سلسلة بإشراف
د. عبد الحسن الحسيني

كورفوازييه و فاليت

أنظمة تشغيل الميكروحاسبات

مبادىء أنظمة التشغيل

Unix - IRMX 86 - CP/M- MS/DOS

ترجمة د. عبد الحسن الحسيني

المؤسسة الجامعية للدراسات والنشر والتوزيع

أنظمة تشغيل الميكروحاسبات

مبادىء أنظمة التشغيل

أنظمة تشغيل الميكروحاسبات

مبادىء أنظمة التشغيل

الطبعة الأولى
١٤١٠هـ - ١٩٩٠م

المؤسسة الجامعية للدراسات والنشر والتوزيع
هاتف ٨٠٢٤٢٨ - ٨٠٢٤٠٧ - ٨٠٢٢٩٦
٣٠١٠٣٠ - ٣١١٣١٠

هذا الكتاب ترجمة :

SYSTEMES D'EXPLOITATION DES MICRO-ORDINATEURS

Concepts et systèmes dominants

Par

M. COURVOISIER- R. VALETTE

تمهيد

تطورت الميكروحاسبات بشكل سريع في السنوات الأخيرة ، حتى إن التصورات والأفكار التي جرى تحديدها ووضعها ، في السنوات القليلة الأخيرة ، لتصميم الحاسبات ذات الحجم الكبير والوسط ، أصبحت قابلة للتطبيق على الميكروحاسبات . وهذه هي تحديداً حالة أنظمة التشغيل التي يمكن أن تكون عائقاً أمام إستعمال حاسبة صغيرة في مستوى مختلف عن لغة المكنة . إضافة لذلك فإن أنظمة تشغيل الميكروحاسبات تحتوي على مُميِّزات خاصة تجعلها أكثر أو أقل سهولة في الاستعمال وأكثر أو أقل فعالية عند العمل . ولكن جميع هذه المميزات ترتكز على مفاهيم وتقنيات مشتركة .

هدف هذا الكتاب هو إعطاء المبادىء الأساسية التي ترتكز عليها جميع أنظمة التشغيل . إضافة لذلك جعل المُستعمل يتأكد من أن نظام التشغيل « يعمل لخدمته » . هكذا ، فالبرامج المُنشأة يجري تنفيذها على شكل من أشكال « المكنة الفرضية » «machine virtuelle» ، وجميع مميزات العتاد يجري إخفاؤها ، فينتج عن ذلك سهولة في الاستعمال ، سرعة في العمل ، وإمكانية « نقل » كبيرة للمناهج التي تصبح مُستقلّة عن المكنات الفعلية .

القسم الأول من هذا الكتاب هو مخصَّص للمفاهيم العامة . بعد فصل مدخلي يُصنّف مختلف أنواع أنظمة التشغيل ومهمّاتها الأساسية ، يجري درس الأواليات التي تشكّل هذه المهام :

ـ المرقاب (monitor) الذي يدير المهام والذاكرة ،

- إدارة عمليات الادخال - الإخراج والسجلات (I/0 and file)
- الملقى مع المستعمل (interface)

سنـدرس مبادىء **هذه الأواليات** مـرتكزين على الأنـظمة المـوجودة حـالياً قيـد الإستعمال .

ويدرس القسم الثاني بعض أنظمة التشغيل المستعملة . ولقد حاولنا دراسة المفاهيم الأساسية وتغطية الأصناف الرئيسية من أنظمة التشغيل المعتمدة للميكروحاسبات . وتم إختيار CP/M وMS/DOS للحاسبات الشخصية ، التي ترتكز على مُعالجات (processor) بطول 8 أو 16 بتة للكلمة و iRMX86 للأنظمة المخصّصة للعمل في الوقت الفعلي ، و UNIX ، وهو النظام الواسع الانتشار في الحقول الجامعية والذي يطمح ليصبح المحطة الرئيسية للحاسبات المرتكزة على المعالجات بطول 32 بتة للكلمة .

الفصل الأول

التقديم العام

1 . مدخل

عـادة ، يُقدَّم مـوقع نـظام التشغيل في النـظام المعلوماتي ضمن مجمـوعة من المستويات أو الطبقات ، كما نرى في الشكل 1.1 .

برنامج تطبيقي
مفسِّر الأوامر والبرامج المُساعدة
نظام التشغيل
ميكروكود
عتاد

شكل 1.1 . تقسيم النظام المعلوماتي الى مستويات

في الـطرفين الأعلى والأسفل يقـع المُستعمل (أو المُستعملين) والمكنـة (أو المكنات) . من الممكن إعتبار الميكروكود وكأنه يُشكِّـل قسمـاً من العتاد لأن ذلـك أصبح تقنية شائعة في الميكروبروسسور . يؤلف نظام التشغيل المفصل الرئيسي بين المنهاج والعتاد .

يقوم دور نظام التشغيل على إدارة أفضل لموارد العتاد التي يتمتع بها المنهاج : مُعـالج ، ذاكـرة (ذاكرات) ، مـداخل ـ مخـارج . . بشكل يتم فيـه تسهيـل عمـل

المُستعمل . لذلك يجب أن يُشكِّل ملقى (interface) بين الموارد الفيزيائية والمستعمل بتقديمه الى هذا الأخير لغة تحكُّم وقيادة (مع مُفسِّر معتمد) وبرامج مُساعِدة . أمام المُستعمل مكنة فرضية سهلة الإستعمال ونموذجية . هذه المكنة الفرضية تأخذ عن كاهل المُستعمل مهمة معرفة المُميِّزات الدقيقة للعتاد الذي تشتغل عليه البرامج التطبيقية ، مثلاً العناوين الفيزيائية لأبواب المداخل ـ المخارج ، السجلات في الذاكرة الحيّة ومميزات النواقل كالاسطوانات المغناطيسية . طبيعة المكنة الفرضية هذه قد تكون مختلفة مما يؤدي إلى ولادة عدة أنواع من أنظمة التشغيل .

2 . مختلف أنواع أنظمة التشغيل

سنعرض هنا مختلف أنواع أنظمة التشغيل ، دون التقيُّد بتلك المُستعملة حالياً في الميكروحاسبات . من الممكن تصنيفها حسب مخططين . الأول يعود الى طريقة استعمال نظام التشغيل :

- أنظمة للاستعمال الشخصي (مُستعمل واحد) .
- أنظمة مُوجَّهة لقيادة ، وضبط العمليات الصناعية .
- أنظمة موجهة لادارة مجامع المعطيات (أنظمة عملياتية) .
- أنظمة للاستعمال العام .

المخطط الثاني يعود إلى صيغة التشغيل الداخلية للحاسب ونظام التشغيل :

- أنظمة مُرتكزة على إخضاع الأعمال (أو أنظمة «batch»)
- أنظمة مُتعدِّدة المهام (multitâches)
- أنظمة متعددة الاجراءات (multitraitements)

1.2 . الأنظمة المرتكزة على إخضاع الأعمال أو الأنظمة « الدُفعية » .

في العامية ، تدعى هذه الأنظمة «batch» أو دُفعية . في هكذا نظام يتم تنفيذ برنامج واحد في كل لحظة مهما يكن حجمه ومدة تنفيذه . فهو يشغل المكنة كاملة طوال مدة دورانه ، وبالتالي يتم معالجة البرامج على التوالي . يجري تخزين البرامج غير المنفَّذة في الذاكرة الخارجية ، وفي بعض الأحيان على شكل « لائحة إنتظار » مع نظام أولوية لها .

هذه الأنظمة كانت الأولى نظراً لبساطة مبدئها. لذا كان من الضروري ، في مراكز الكومبيوتر القديمة ، وضع حدّ لمدة تنفيذ البرنامج منذ البداية ، لتفادي الحصول على فواتير مُدهشة .

إن بساطة الأنظمة المُرتكزة على إخضاع الأعمال أو المُعالجة « الدفعية » ، يُدفع ثمنها على حساب المتانة عندما يجب تقسيم الحاسب بين عدة مُستعملين . هكذا ، فليس بإمكان المستعملين أن يتدخلوا في برامجهم التي تدور بشكل مُستقل ، مع مدة جواب غير متوقعة لأنها تتعلّق بعدد الأعمال الخاضعة . وعلى العكس ، في حالة الحاسبات الشخصية ، نرى فائدة هذه الأنظمة لأن المُستعمل هو المسؤول الأساسي عن مكنته ؛ والتفاعل هو ممكن ولكن بشرط أن يتم توقع ذلك في البرامج .

2.2 . الأنظمة المتعددة المهام

دور هذه الأنظمة هو السماح بتقسيم عمل المُعالج بين عدة برامج ، التي من وجهة نظر المستعمل ، تدور في نفس الوقت .

يرتكز مفهوم تزامن التنفيذ هذا ، على إستعمال أفضل للمعالج المركزي . يتم تخصيص المعالج للبرامج حسب أولويتها واحتياجاتها بواسطة قواعد تنظيم معقدة . هذا المفهوم ، الذي يُدعى عادة برمجة متعدِّدة (multiprogrammation) . يسمح بتواجد برامج مستقلة تماماً أو مُتحدة لتنفيذ هدف مشترك وذلك بتبادل المعلومات وبتقسيم المعطيات . ويسمح نظام التشغيل بشكل عام بإدارة الموارد المُوزّعة (ذاكرة ، سجلات ، طابعة ، الخ) ، ويسهر على تماسك إستعمال المعطيات المشتركة وتبادل الرسائل .

تدعى سلسلة التعليمات الجارية للتنفيذ في إطار معين مهمة أو مُعالَجة (pro- cessus, tâche) ؛ أي من وجهة نظر ديناميكية ، تشكّل برنامجاً معيناً . والعمل التطبيقي يمكن أن يتألف من عدة مهام منشأة ، حيّة وهامدة حسب تطورها . هذا التنظيم هو أكثر بساطة من التطبيق الموضوع في العمل بواسطة برنامج واحد . وحسب حقل التطبيقات ، جرى تطوير أنظمة تشغيل مختلفة من خلال مفهوم أساسي . سنقوم هنا بتفصيل بعض التصوّرات الممكنة .

أ ـ الأنظمة في الوقت الفعلي (real time system)

الميزة الأساسية لأنظمة الوقت الفعلي هي في السماح بالمعالجة في الوقت الفعلي للحوادث ، أي بتأمين جواب في مدة معينة . هذا الإلزام المتعلّق بمدة التنفيذ يؤدي إلى تصنيف الأعمال المطلوبة حسب مختلف مستويات الأفضلية ، تلك التي يكون فيها الوقت عبارة عن إلزام فعلي أو شرط أساسي تكون بأفضلية كبرى . في الحالة الأسهل ، نلتقي مجموعتين من هذه الأعمال : الأعمال المطلوب تنفيذها في

الوقت الفعلي (foreground programs) والأعمال بدون أولوية (background programs) . هذه الفئة الأخيرة تتعلّق بكل ما يمكن أن يحدث في الأوقات الفارغة والميتة لتنفيذها كما يجري عادة بالنسبة لبعض الحسابات الإحصائية ، إصدار بعض النتائج ، النشرات ، الخ .

نرى فائدة إستعمال نظام متعدد المهام في هذا المفهوم ، حيث يناسب كل عمل مطلوب تنفيذه مهمة معينة بأولوية معينة ، وهذه المهمة يمكن تعليق دورانها في أي لحظة للسماح بتنفيذ مهمة أخرى بأولوية أكبر .

أغلب أنظمة الوقت الفعلي مُوجَّهة للعمل في الحقل الصناعي ؛ الحاسب أو الحاسبات تكون مرتبطة بعملية فيزيائية .

مثلاً ، نجد أنظمة التحكُّم بالعمليات الصناعية (كيمياء ، بترول ، صناعة . . .) ، حيث تُستعمل الحاسبات لتنظيم الانتاج أي أتمتة الانتاج . تكامل مختلف مستويات الأتمتة ، من بداية التحكّم المركزي حتى برمجة الإنتاج يصبح ممكناً بواسطة وسائل المعلوماتية . هذا التكامل هو أحد مركّبات الانتاج . والحاسبات التي تقع على عاتقها مُهمّة التحكم المركزي تتفاعل دوماً وبشكل ثابت مع العملية الصناعية بواسطة لواقط (capteurs) أو مُفاعلات . إختلاف العمليات الصناعية يجعل من مدة الجواب مختلفة من عدة مللثوان (كالإنسان الآلي المتحرِّك مثلاً) الى عدة دقائق (صناعة السيراميك ، والتحكُّم بالأفران) ، إلى عدة ساعات (العمليات البيولوجية)

نجد أيضاً أنظمة الإتصال حيث مهمة الحاسب تقوم على إرسال ومعالجة الرسائل . تتم عملية الإرسال بواسطة خطوط أو شبكات هرتزية وتُقطّع الرسائل في حزم ، مكوّدة ، متراصة ، مُضمَّنة وتُرسل بسرعة كبيرة . ويتمّ إعادة تشكيل الرسائل وإختبار التصحيح في الوقت الفعلي . ثوابت الوقت تكون قصيرة جداً ، وأقل من مللثانية ، وبالتالي تستدعي عادة إستعمال عتاد متخصِّص .

هناك حقل آخر لاستعمال أنظمة الوقت الفعلي يتعلّق بالتطبيقات العسكرية حيث نجد عدة مسائل مركّزة حول الإتصالات والقيادة من بعيد وبدرجة من التعقيد تكبر مع الشروط القاسية للسرية التي تستدعي إستعمال مناهج وعتاد إضافي .

وفي النهاية تجدر الإشارة إلى أنظمة التحاور والمحاكاة التي تهدف إلى إيجاد وسيلة لاستبدال النظام الفيزيائي بواسطة نظام معلوماتي يؤدي نفس العمل وتقديمه إلى

المستعمل . هكذا أنظمة تتطلب ، في أغلب الأحيان ، نظاماً للرسم البياني (graphic system) بإنجازات دقيقة كما يجري مثلاً في أنظمة قيادة الطيران .

ب ـ أنظمة مُتعدِّدة ـ المستعملين (multiuser system)

الاختلاف الرئيسي بين أنظمة تعدُّد المستعملين وأنظمة الوقت الفعلي للتحكم بالعمليات الصناعية يكمن في كون الأعمال هي مستقلة في الأنظمة الأولى، لأن كلاً منها يتعلَّق بمستعمل مختلف ، بينما تكون المهام في أنظمة « الوقت الفعلي » متماسكة تتساعد جميعها على تنفيذ التحكُّم .

يتصور كل مستعمل بأنه لوحده يستعمل المكنة . نظام التشغيل « مُتعدِّد المستعملين » يجب أن يدير مجموع موارد المكنة (معالج ، ذاكرة ، إسطوانات . .) بتأمين حماية للمستعملين بشكل يؤمن تفادي التدمير اللاإرادي أو عن سابق تصوُّر للمعطيات الخاصة بأحد المستعملين . إضافة إلى ذلك فإن إدارة الموارد هذه يجب أن تكون منصفة كي يتم تأمين تعايش بين أعمال ومهام المستعملين على المكنة خصوصاً فيما يختص بعملية تخصيص المعالج ، هذه القسمة المتساوية يجري حلُّها بتخصيص كل مُستعمل بمدة زمنية معينة وبشكل دوري مما يؤدي إلى أنظمة الوقت المقسَّم (time sharing systems) .

يؤدّي تقسيم الوقت إلى قطع متساوية لجميع الأعمال إلى الإختلاف بين نظام تعدُّد المستعملين مع نظام التحكم بالعمليات الصناعية . هكذا ، ففي الحالة الأخيرة يجب ألا تكون الآماد الزمنية متساوية ، لأنه على العكس فإن الأعمال المستعجلة يجب أن تشغل المعالج بالكامل وأن تعلَّق الأعمال الأقل إستعجالاً طوال الوقت اللازم لتنفيذها .

ج ـ الأنظمة التصالحية (transactionels)

أنظمة التشغيل متعددة المستعملين هي أيضاً أنظمة متعدِّدة البلوغ (multi accès) . لذلك فإننا نخلط بينها وبين الأنظمة متعددة ـ الأدوات الطرفية (multi-terminals) التي تختلف فعلياً عنها في بعض الأحيان . هكذا ، فهذه الأخيرة ، متعددة ـ الأدوات الطرفية ، ترتكز على استعمال برنامج مشترك بين عدة مستعملين بالتزامن . نطاق الإستعمال المعروف يتعلق بالأنظمة التصالحية حيث نواجه مُشكلة البلوغ المتزامن لمجمع مشترك من المعطيات (data base) . في هذه الأنظمة ـ أنظمة حجز المقاعد على الطائرات مثلاً ـ يجب أن يؤمن نظام التشغيل وبشكل

مستمر تماسك وتكامل وآمان مجمع المعطيات الذي يستعمله عدة مستعملين .

3.2 . الأنظمة التي تسمح بالمعالجة ـ المتعدد: (multitraitement)

في هذا النوع من الأنظمة ، التزامن الفعلي لتنفيذ عدة برامج في وقت واحد هو ممكن ، بسبب وجود عدة مُعالجات . مهمة أنظمة التشغيل هي في ترتيب المعالجة على عدة مُعالجات يمكن أن تكون متشابهة (أي بإمكانيـة مُعالجـة متقاربـة) أو غير متشابهة . التشغيل يتم حسب مبدأ تقسيم الحِمل(load sharing) ، أي يُمكن للبرنامج وخلال مدة دورانه أن يتم تنفيذه على التوالي على عدة مُعالجات وذلك في كل مرّة يتم فيها تعليق تنفيذه ومعاودته . ويتمّ تخصيص المُعالجات للبرامج بشكل يوازن على أفضل صورة وفي كل لحظة شحن هذه المعالجات .

مواصفات المُعالجات هي معروفة ، ومن الممكن تعريف خوارزميات الترتيب الفعالة . هكذا ، يمكن وضع المسألة في مستوى مُعالجات الإدخال ـ الإخراج لأنّ مدّة التبادل والجواب هي صعبة التوقع بشكل مسبق . يُمكن أن نحتاج من جديد للمعالجة المتعددة .

في هـذا النــوع من الأنظمـة ، ومهما تكن مـواصفات المعـالجات متشـابهة (أنـظمة تـدعى متعـددة ـ المُعـالجـات) أو مهمـا تكن درجـة تخصيصهـا بـالمهـام (مُعالجات لمعالجة الإشارات الخاصة) ، فإن نظام التشغيل يقع على مُعالج مركزي يشرف على عمل المُعالجات الأخرى .

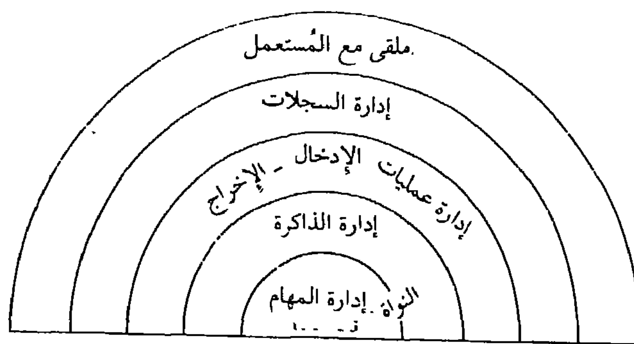
وعلى العكس ، توجد أنظمة ، حيث المُعالجات لا تتمتـع بمميزات خـاصّـة وحسب ، ولكنها إضافة لذلك تمتاز ببعض الاستقلالية في إتخاذ بعض القرارات (أي تتمتع بذكاء مركزي) . في هذه الحالة ، على كل مُعالج يوجد نظام تشغيـل ومكان كافٍ في الذاكرة المركزية لكي يستطيع تشكيل حاسب وحيد لنفسه . التوصيلات بين الحاسبات تكون بشكل عام بطبيعة مُتسلسلة مما يؤدي إلى تعريف إتفاق للإتصالات المركبة فيما بينها . نحصل عند ذلك على طبقة أنظمة المعلوماتية الموزعة . الشبكات المركزية هي مثال على ذلك . يجب أن نضيف إلى الأعمال العادية لنظام التشغيل ، التي هي من النوع متعدد المهـام ، الإجراءات التي تسمـح بتأمين تمـاسك وتكـامل المعطيات المُوزعة . هكذا ، ففي هذا النـوع من الأنظمـة ، لا يوجـد مفهوم للوقت الموحد وتزامن المعطيـات يمكن أن يكون كـاملاً (تـوازٍ مُتكامـل وفعلي وليس فقط ظاهري) .

وتجري الآن دراسات عديدة على الأنظمة الموزعة التي تتطور بسرعة بسبب زهد ثمن الحاسبات الصغيرة وقوتها التي ما زالت تتصاعد باستمرار .

3 . مهام نظام التشغيل

كما رأينا في بداية هذا الفصل ، فإن نظام التشغيل يُشكِّل الملقى بين العتاد ومناهج المستعمل . لكي يكون للمستعمل نظرة مُبسَّطة للمكنة الفرضية وكي يستطيع إستعمال العتاد بشكل عملياتي تُنظَّم أنظمة التشغيل في مستويات أو طبقات . وهذا صحيح بالنسبة لأنظمة تشغيل الميكروحاسبات لأن تطوير أنظمة التشغيل يصبح أكثر سهولة كي تتلاءم مع التطور السريع في عتاد الميكروحاسبات ، وكي يتأمّن التوافق التصاعدي فيما بينها .

ولو وضعنا جانباً أنظمة التشغيل « موحَّدة ـ المُستعمل » المشتقة من الأنظمة التي ترتكز على إخضاع الأعمال (Batch) أو الأنظمة « الدفعية » التي تعمل عليها غالباً الميكروحاسبات ، فإن أنظمة التشغيل قد جرى إنشاؤها من خلال نواة (Kernel أو nucleus) تؤمن إجراء بعض المهام الأساسية . عدد المهام التي تؤمن تنفيذها النواة تتعلَّق بالتقسيم المُختار من قِبل المُصمِّم . سنختار التقسيم المُتمثِّل على الشكل 2.1 . . . الأنظمة الأكثر سهولة ، وكونها غير متعددة المهام ، ولا تحتوي على برامج لإدارة المهام ولا على برامج لإدارة الذاكرة .



شكل 2.1 : مستويات نظام التشغيل

1.3 . النواة

النواة ، وكما يدل الشكل 1.2 ، تحتوي على مهمتين رئيسيتين : إدارة الذاكرة

وإدارة المهام ، وتُدعى غالباً بالمرقاب (monitor) .

يتضمّن مُنظِّم المهام في نظام التشغيل على الأقل المهام الثلاث التالية :

- الأخذ بالحسبان ، ومعالجة الإنقطاعات (ميقت الوقت الحقيقي عبارة عن إنقطاع بأولوية كبرى تدار في هذا المستوى) .
- تنظيم المهام حسب القواعد الأكثر ملاءمة بواسطة مُنظِّم للمهام (sheduter , dispatcher) .
- معالجة المهام بواسطة مجموعة إجراءات .

هذه الإجراءات تسمح بإطلاق ، عملية التزامن (Synchronisation) ، وإنهاء تنفيذ المهام . ومن الممكن أيضاً ، في بعض الأحيان ، نقل المعلومات من مهمة إلى أخرى (تبادل الرسائل) . هذه المهام الأخيرة هي من إختصاص مُنظِّم المهام ومُنظِّم الذاكرة في نفس الوقت: تطلق حرية مُنظِّم المهام عندما يتم دوران العمل التطبيقي .

ويكتب بطريقة مثلى ، وبشكل عام يكون في لغة المؤول (assembler) ، ويرتبط بشكل كبير بالعتاد .

تقوم مهمة مُنظِّم الذاكرة أو برنامج إدارة الذاكرة على تخصيص مناطق أو حيزات من الذاكرة الى مختلف الأعمال والمهام .

وفي لحظة معينة ، يسهر هذا البرنامج على عدم بلوغ أي من المهام أو الأعمال للمكان ، أو الموقع من الذاكرة المُخصَّص لمهمة أخرى أو عملٍ آخر . مُنظِّم الذاكرة يصبح شديد الأهمية عند وضع الذاكرة الفرضية في العمل (Virtual memory) بواسطة عمليات التصفيح (paging) والتقسيم (segmantation) للذاكرة . هذا التصوُّر كان حتى الآن محصوراً بالمكنات ذات الفعالية الكبيرة (حاسبات وسطية (mini) أو حاسبات كبيرة) ، ولكن الإتجاه الجديد في تطوير الميكروبروسسور من نوع 16 أو 32 بتة للكلمة قد أخذ بعين الإعتبار إستعمال نظام التشغيل من نوع UNIX الذي يحتوي على الذاكرة الفرضية .

مُنظِّم الذاكرة هو أيضاً مطلوب أكثر من مُنظِّم الأعمال ، لذلك فهناك محاولة حالية تقوم على تخزين عمله في دارات تكاملية مُتخصِّصة تدعى (Memory MMU Management unit) . وهذه هي الحالة لدى الميكروبروسسور من نوع 32 بتة

(national NS 32032 ، (ATT) WE 32100 ، (motorola) MC68020 semiconductor) و80386 (intel) التي تجمع ما بين عمليات التقسيم والتصفيح للذاكرة .

2.3 إدارة عمليات الإدخال ـ الإخراج (input-output)

جرت مُعالجة هذه العملية وحسب الحالة في التطبيقات التي تتم في الوقت الفعلي بواسطة مختلف الأدوات المحيطية والفروقات بين مميزاتها . ولكن أنظمة التشغيل الحديثة تسمح بمعالجة الأدوات المحيطية كوحدة منطقية ، أو قنوات (channels) ، بمُميزات فيزيائية ظاهرة للمُستعمل . وهذا يتم على حساب خسارة ضعيفة في الأداء .

يتم وصف الوحدات المنطقية بشكل موحّد (واصف الوحدات) ، ويجري تقديم معطياتها على شكل متغيرات وسيطية . إستعمال هذه الوحدات يتم بواسطة مُنظّم لهذه الوحدات (handler) ، ويُحاول المستعملون إجراء طلبات لها توضع في لائحة إنتظار . ولجعل عمل مُنظّم وحدات الإدخال ـ الإخراج أكثر فعالية تجري إضافة دارىء (buffer) له لتخزين الطلبات . وفي النهاية فإنّ وحدات الإدخال ـ الإخراج الأكثر إستعمالاً يُمكن أن تُنظّم بواسطة برامج خاصة تدعى **Spooler** قادرة على إجراء معالجة متوازية ومتماسكة للطلبات الآتية من عدة مستعملين . تختلف هذه التقنيات تتمتع بأثر شبيه بمفهوم الذاكرة الفرضية ومن هنا مفهوم الأداة الطرفية الفرضية (Virtuel terminal) . يرى المستعمل وحدات الإدخال ـ الإخراج بشكل نموذجي مُعتقداً بأنه المُستعمل الوحيد الذي يعمل على الحاسب .

3.3 . إدارة السجلات

السجلات هي عبارة عن مجموعات من المعلومات تسمح بتخزين :

- نظام التشغيل نفسه .
- برامج ومعطيات المُستعمل .
- برامج مُساعدة (ربيدة أو مكتبة البرامج) .
- منقحات ، مصرّفات ، . . .

بعض المجموعات يمكن أن تكون راكنة (résident) (أي دائماً موجودة في الذاكرة الحيّة) ، وهذه هي الحالة تحديداً بالنسبة لأقسام من نظام التشغيل الأكثر إستعمالاً (نواة خاصة) ، البعض الآخر يقع في الذاكرة الخارجية (إسطوانة مرنة أو

صلبة حسب الميكروحاسب المستعمل) .

كون تركيبة السجلات محدَّدة (أبعاد الفدرات) ، تمتاز مهمة إدارة السجلات بدور يتعلَّق بتنظيم البلوغ . يستعمل لذلك فهرس أو قائمة (Directory) بأسماء السجلات . وفي بعض السجلات ، وبدلاً من وجود فهرس موحد لجميع السجلّات ، من الممكن تنظيمها على شكل تراتبية (خوارزمية hiérarchie) من القوائم المبلوغة بواسطة القائمة الأساسية . هذه القوائم تسمح بإجراء تناسب بين العنوان المنطقي للسجل (اسمه) ، وعنوانه الفيزيائي (مثلاً رقم المسار والقطاع الدائري المناسب لبداية السجل على الاسطوانة) .

يمكن أن تكون السجلات ببلوغ متتالٍ (نبلغ دائماً المعلومات بواسطة قراءة السجل من البداية) أو عشوائي (بالإمكان قراءة أو تعديل قسم من السجل وذلك ببلوغه مباشرة) أو بحجم ثابت أو متغيِّر .

حسب مختلف هذه الحالات ، فإن إدارة السجل ستستعمل تقنيات أكثر أو أقل تعقيداً . مثلاً ، لإدارة سجلات ببلوغ عشوائي وبحجم متغيِّر ، فإن برنامج تنظيم السجلات سيستعمل جدولاً بالمؤشرات (index table) .

بعض السجلات يمكن أن تكون مقسومة بين عدة مستعملين ويجب على مُنظِّم السجلات (البرنامج الذي يدير السجلات) أن يتحقَّق من حقوق البلوغ قبل السماح باستعمالها . مهام مُنظِّم السجلات المبلوغة مباشرة من المستعملين هي فتح السجل (لقراءته أو لكتابته) وإغلاقه (إستيفاء يومي للقوائم قبل تعديلها) .

3 .4. الإتصال مع المستعمل

هذا المستوى الأخير في نظام التشغيل يسمح للمستعمل بالحوار الفعّال مع نظام التشغيل . وبواسطة هذا الإتصال يستطيع المستعمل أن يُنشيء برنامجه التطبيقي ، أي أن يبني سجلات برامج مصدرية تحت إشراف المُنقِّح (editor) ، أن يستدعي المصرِّف (compiler) ومنقِّح الأربطة (link editor) ، أن يخطِّط النظام حسب وحدات الإدخال ـ الإخراج المُستعملة ، أن يطلق البرنامج التطبيقي وأن يتابع تنفيذه بواسطة برامج مُساعِدة للتطوير والمُساعدة .

لتحريك جميع هذه المهام ، يرتكز ملقى المُستعمل على لغة تحكُّم ، مفسِّر للأوامر ، وفي النهاية على مُنقِّح للأسطر يسمح بإدخال الأوامر وتعديلها في حالة الأخطاء .

3 . 5. خاتمة

في أنظمة التشغيل الخاصة بالميكروحاسبات ، مختلف المستويـات التي قمنا بتحديدها جرى تـطويرهـا حسب حقل الإستعمـال المرئي للحـاسب . إنّ زيادة قـوة الميكروحاسبات أدت إلى أن تصبح أنـظمة التشغيـل متعددة المهـام (multitasking) قابلة للتعميم . سنقوم لاحقاً ، قبل تفصيل بعض أنظمـة التشغيل الـرئيسية ، بشـرح الـمفاهيم المتعلّـقة بمختلف مستويات نظام التشغيل بشكل مفصّـل وكاف .

الفصل الثاني

المرقاب (الوقت الفعلي)

نواة نظام التشغيل

1 . مدخل

يقوم دور المرقاب على إدارة المُعالج والذاكرة . ويعتبر ذا أهمية كبرى في نظام التشغيل متعدد المهام أكثر منه بالنسبة للأنظمة المُتخصِّصة في التحكم بالعمليات الصناعية أو الأنظمة متعددة - المستعملين . الحالة الأولى تؤدي إلى وجود عدة مهام تتجمع لتنفيذ هدف مشترك ، الثانيه تناسب مهام عملياتية مستقلة ولكنها تستعمل موارد مشتركة .

وفي الحالتين ، يجب أن نقوم بتعريف تقنية تسمح باستعمال أفضل للمعالج والذاكرة . هذه التقنيات تعمل على تقسيم الوقت و/ أو على تقسيم مساحة الذاكرة .

2 . إدارة المهام

قبل البدء بمعالجة إدارة المهام ، سنقوم بتحديد مفهوم « المهمة » (task) كما يراها المُستعمل الذي يرغب ببرمجة عمل تطبيقي معيّن للتحكُّم بالمعالجة الصناعية بواسطة نظام تشغيل في الوقت الفعلي .

2 .1. تركيب العمل التطبيقي بواسطة مهام

المرحلة الأولى تقوم على تعريف العمل التطبيقي . وكي نقوم بذلك بشكل جيد ، نستعمل المفهوم الإنحداري الذي بواسطته نبدأ بتعريف الأعمال الرئيسية والتفاعل فيما بينها . هذه الأعمال يمكن أن تعتبر وكأنها عبارة عن مجموعة مهام يجب القيام بها لإنهاء نظام التحكُّم . تفصيل العمليات ، أي المضمون أو جسم المهام يمكن أن يحدد من جهة ويجب أن لا يهتم بموضوع التفاعل والعلاقات بين مختلف

الأعمال الرئيسية كي يكون الإجراء فعالاً .

مثلاً : التحكُّم بنظام لانتاج الورق

يتعلَّق ذلك بتصوُّر نظام يسمح بضبط درجة الحرارة لكدسة من الورق ، إضافة إلى ضغط لفَّة الورق للحصول على سماكة منتظمة للأوراق ، يجب أن يكون هناك علاقة مع المؤشر (operator) الذي يرغب بتعديل إشارات الحرارة والضغط وطلب عرض حالة العمل .

في هذا المثل ، يجب تعريف عدد من المهام المختلفة :

- قراءة وتفسير أوامر المؤثر ،
- تغيير متغيرات الضبط ،
- ضبط الحرارة ،
- ضبط الضغط ،
- عرض على القنصلة (console) .

بعض هذه العمليات هي مستقلة ويمكن أن تتم بشكل متزامن . البعض يمتاز بأولوية كبيرة ولا يجب أن يُقطع خلال مدة طويلة ، وهذه هي حالة الضبط . من المهم إذاً ، كي تتم عملية الضبط بشكل جيد ، تحديد العلاقات والتفاعل فيما بينها .

الدوران المتتالي المتزامن هو ممكن . وسيقوم على ربط هذه العمليات في برنامج موحد على الشكل التالي (« ؛ » ، يعني على التوالي) :

- قراءة وتفسير أوامر المؤثر ؛
- تعديل المتغيرات (إحتمالاً) ؛
- ضبط الحرارة ؛
- عرض الحرارة ؛
- تعديل المتغيرات (إحتمالاً)؛
- ضبط الضغط ؛
- عرض للضغط ؛
- عودة إلى البداية .

هكذا حلٌ هو مُكْلِف لجهة الوقت لأن الحاسب يقوم بفحص ، وفي كل دورة زمنية ، ما إذا كان يوجد أوامر جديدة خاصة بالمؤثر . نقوم بالعرض في كل دورة .

هذه العمليات يجب أن تتم بطريقة لا تزامنية حسب رغبة المؤثر العامل .

هذا ما يثبت الفائدة من إستعمال المرقاب في نظام التشغيل «مُتعدِّد المهام » من أجل وضع الإدارة الديناميكية للأعمال الموضوعة في العمل . من جهة أخرى ، هذا العمل يجب أن يكون مسبوقاً بنموذج شكلي وعام بواسطة إحدى الوسائط (شبكة بتري (petri) ، مثلاً) القادرة على تحديد التطورات المتزامنة (التوازي parallélisme) واللاتزامنية لهذه الأعمال إضافة إلى التفاعل فيما بينها .

الفائدة في هذا النموذج الشكلي العام والمتعدِّد يكمن في :

- السماح ، عندما نستعمل وسيلة شكلية ، بالتأكيد على صلاحية مخطط التزامن والعمل التطبيقي والتبرير المجتزأ لتقسيم المهام (للإنهاء بواسطة تحليل المعطيات المقسَّمة بين المهام) ،
- السماح أيضاً بوضع التفاعل بين المهام مباشرة في العمل ، أو تقريباً بشكل مباشر .

2.2 . مفهوم المهمة

أثناء مرحلة تصوّر العمل التطبيقي ، تكون المهام عبارة عن عمليات للتنفيذ . مفهوم « المهمة » هو إذاً عبارة عن مفهوم شكلي نسبياً يُحدَّد قليلاً قليلاً . هذا المفهوم يصبح شكلياً عندما نصل إلى مرحلة البدء في تنفيذ المهام تحت نظام التشغيل (وقت فعلي متعدِّد المهام) .

تتألف المهمة إذاً من جزء من البرنامج ، يُدعى جسم المهمة ، ومن معطيات عمل تُمثِّل إما معطيات خاصة أو معطيات مقسَّمة بين مهام أخرى وفي النهاية من « واصف » للمهمة .

جسم المهمة هو عبارة عن سلسلة من التعليمات القابلة للتنفيذ ، بينما سيتم تفصيل معنى « واصف المهمة » في الفقرة 3.2.2 .

في بعض الأحيان ، نستعمل مفهوماً أكثر إتساعاً من المفهوم السابق ، هو مفهوم « العملية » processus ، وبالأخص بالنسبة للمهام الداخلية لنظام التشغيل أي المهام التي ينشئها هذا النظام ديناميكياً لإجراء بعض الخدمات للمستعملين . للعملية وسيلة بلوغ مُباشرة لجميع المعطيات وتختص بواصف مبسَّط .

2.2.1 . حالات المهمة

من الناحية العملياتية ، أي من وجهة نظر المستعمل ، تكون المهمة إما في طور

التنفيذ وإما متوقفة لأن معطيات العمل ليست جاهزة أو لأن تنفيذها ليس مفيداً في لحظته .

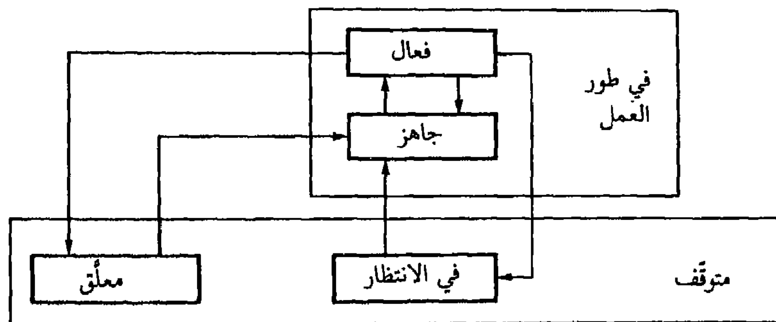
من وجهة نظر المُعالج ، أي من وجهة نظر نظام التشغيل للحاسب ، تمرّ المهمة في أربع حالات ، هذه الحالات يمكن أن تكون :

- فعالة : المهمة هي في طور التنفيذ بواسطة المعالج .
- جاهزة : المهمة هي جاهزة للتنفيذ وتتمتع لهذه الغاية بجميع الموارد الضرورية ما عدا المُعالج ؛ المهمة الفعالة هي تلك التي تمتاز بالأولوية الكبرى .
- في الانتظار : المهمة هي في انتظار إحدى الحوادث ؛ إذا كانت تفتقد لأحد الموارد (طابعة ، مساحة من الذاكرة ؤ. .) فهي تنتظر إنقطاعا فيزيائيا أو رسالة تشير إليها بأن المورد المطلوب تم تخصيصه لها .
- معلّقة : المهمة هي معلّقة عندما لا يرغب المستعمل بأن يكون في موقع تنافس مع غيره من المستعملين على المُعالج ، وهي أيضاً معلّقة في الحالة التي لا تكون فيها مفيدة (مهمة تصفير وإعداد مثلاً) .

الحالات والانتقال من حالة إلى أخرى هي ممثلة على المخطط في الشكل 2 . 1.

سنُضيف إلى هذه الحالات لائحة تحتوي على أسماء المهام الموجودة في الحالات المناسبة :

- لائحة بالمهام الجاهزة ، المهمة الأولى من هذه اللائحة هي المهمة الفعالة .
- لائحة بالمهام في حالة الانتظار ،
- لائحة بالمهام المعلّقة .

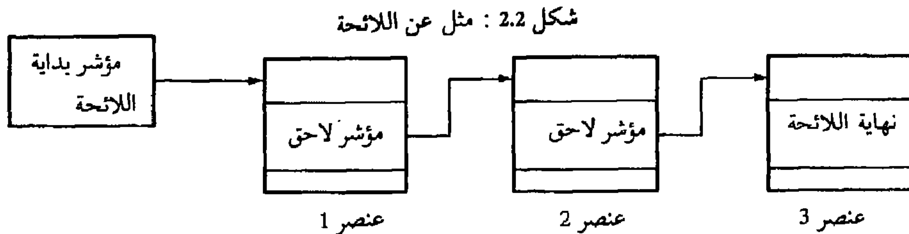


شكل 1.2 . حالات المهمّة

كل مهمة من النظام تتواجد على لائحة وحيدة فقط ، ترتيب المهام في هذه اللوائح يتعلّق في نفس الوقت باللحظة التي تم وضعها فيها وبأولويتها . المهام ذات الأولوية العليا توضع في رأس اللائحة . وعندما تتغير حالة المهمة ، ليس من الضروري أن توضع في طرف اللائحة ، بل يتم إدخالها في موقع محدَّد ، وبالتحديد بعد المهام التي تتمتع بأولوية أعلى من أولويتها .

وكي نجعل إجراءات الإدخال أكثر سهولة وسرعة ، فإن هذه اللوائح تتشكّل من عناصر موزعة ومرتبطة بحلقات بواسطة مؤشرات (pointers) . المؤشر عبارة عن متحولة خاصة ، التي وبدلاً من أن تحتوي مباشرة على معلومات معينة ، فهي تحتوي على عنوان فدرة من الذاكرة حيث توجد المعلومات . الشكل 2.2 يظهر إنشاء اللائحة بواسطة المؤشرات . هذه الأوالية سيتم تفصيلها في 3.2.2 عندما سنحدِّد واصف المهام .

شكل 2.2 : مثل عن اللائحة



2.2.2 . تعديل حالات المهمة

عند تغيير حالة المهمة بإمكاننا أن نلتقي إحدى الحالات التالية (إحتمالاً أكثر من واحدة في نفس الوقت) :

1 ـ المهمة الفعالة ينقطع وجودها

- بسبب حادثة خارجة عن المهمة (إنقطاع أولوي ، إنقطاع لميقت الوقت الفعلي) التي تتطلب تنفيذ مهمة مرتبطة بهذه الحادثة وأكثر أولوية من المهمة الفعالة ؛ هذه المهمة ذات الأولوية هي إذاً معلّقة في رأس لائحة المهام الجاهزة ، وقبل المهمة التي كانت فعالة .
- بسبب كون الوقت المخصَّص للمهمة الفعالة قد إنتهى ولوجود مهام أخرى جاهزة وبأولوية تساوي تلك الخاصة بالمهمة الفعالة ؛ فإن المهام الجاهزة ذات الأولوية نفسها سيتم تنفيذها بشكل مشبك ؛ يجري دوران للمهام ذات الأولوية نفسها في لائحة المهام الجاهزة ؛ هذه الأوالية تدعى « قطعة وقت tranche de temps »

ومستعملة في أنظمة التشغيل متعددة ـ المستعملين (multi-users) كالنظام UNIX ولا تستعمل في الأنظمة التي تعمل في الوقت الفعلي (real time) لقيادة العمليات الصناعية .

ـ بسبب الحادثة التي تثيرها المهمة الفعالة نفسها كالرسالة المرسلة بواسطة مهمة أقل أولوية كانت موجودة على لائحة المهام المُنتظرة (تحديداً في هذه الرسالة) ؛ فالمهمة المُسْتقبِلة للرسالة سيتم إدخالها في رأس لائحة المهام الجاهزة قبل المهمة التي كانت فعالة .

ـ لأن المهمة الفعالة قد وُضِعت في إنتظار حادثة معينة؛ فهي إذاً مُنتزعة من لائحة المهام الجاهزة ليتم إدخالها في لائحة المهام « في الانتظار » ؛ المهمة الموجودة في رأس لائحة المهام الجاهزة تصبح إذاً فعالة ، هذه الحادثة يمكن أن تكون رسالة ، انقطاعا، أو اشارة تقول إلى أن بعض الوّقت (تأخير) قد مرَّ .

ـ ولأن المهمة الفعالة قد عُلِّقت ؛ فهي إذاً ستخرج من لائحة المهام الجاهزة لتذهب الى لائحة المهام المُعلَّقة .

2 ـ المهمة المُنتظرة ستدخل في لائحة المهام الجاهزة حسب أولويتها :

ـ بسبب حادثة خارجية (إنقطاع) ،

ـ بسبب حادثة اتية من مهمة فعالة (رسالة) ،

ـ لأن مدة الانتظار قد إنتهت (سبقت الوقت الفعلي) .

3 ـ المهمة المُعلَّقة هي جاهزة ويتم إدخالها في لائحة المهام الجاهزة :

ـ لأن المهمة الفعَّالة والعاملة قد طلبتها في نظام التشغيل .

3.2.2 . تركيبة المعطيات المرتبطة بالمهمة

لقد رأينا ان المهمة هي عبارة عن سلسلة من التعليمات التي تعمل على معطيات معنية . المعطيات الخاصة بالمهمة لا يمكن أن تكون مبلوغة بواسطة المهام الأخرى ، على العكس فإن المعطيات الموزعة يمكن أن تكون مبلوغة من المهام الأخرى . هكذا ، إضافة الى هذه المعطيات المعروفة والمحدّدة بواسطة مبرمج العمل التطبيقي ، يوجد معطيات أخرى مضافة إلى المهمة : إنّها المعطيات التي ينشئها نظام التشغيل والتي وحده يمكنه بلوغها .

تتألف هذه المعطيات من العنصرين التاليين :

- مكدس ،
- واصف للمهمة .

مهمة المكدس هي تخزين مجموعات المعطيات الضرورية لاعادة تشكيل حالة المهمة وغير الموجودة في ذاكرة الحاسب . يتعلّق ذلك بشكل أساسي بقيم المراصف الداخلية للمعالج كعداد البرنامج (Program counter, Compteur ordinal) الموجود في المعالج المركزي للحاسب . وعندما لا تعود المهمة فعّالة وعاملة ، يجب إجراء عملية حفظ لنصها وذلك بترتيب جميع هذه المعطيات في المكدس . وعندما تعود المهمة إلى حالة الفعالية يجري إستخراج النّص من المكدس بشكل اوتوماتيكي .

يحتوي واصف المهمة على المتغيرات المستعملة بواسطة المرقاب لإدارة متغيرات حالات المهام وتشكيل اللوائح ، عدد المتغيرات الموجود في الواصف (descriptor) يتعلّق بتعقيد نظام التشغيل . وبشكل عام ، نجد العناصر الأساسية التالية :

- عنوان المكدس ،
- طول المكدس ،
- مؤشر المكدس ،
- مؤشر التأخير (الابطاء) .
- مؤشر اللائحة ،
- التأخير (delay) ،
- الحالة ،
- الأولوية .

المتغيرات الثلاثة الأولى تسمح بتحديد وتعريف المكدس في الذاكرة . مؤشر الإبطاء يؤمن ربط واصفات المهام في الانتظار (نجده في عنوان واصف المهمة التالية في هذه اللائحة) . مؤشر اللائحة يلعب نفس الدور في حالة لائحة المهام الجاهزة وتلك المعلّقة ؛ لا يوجد أي إبهام أو خلط لأن المهمة لا يمكن أن تكون في نفس الوقت جاهزة ومعلّقة .

يمكن أن نتساءل لماذا لائحة المهام في الانتظار تجري معالجتها على حدة . هذا يأتي من كون أغلب برامج المرقاب تتوقع إعتماد وقت أطول للإنتظار (من

المحتمل أن يكون غير مُحدَّد) مهما تكن الحادثة المُتوقَّعة . تُسجَّل مدة الانتظار هذه في مُتغيِّر التأخير وتنقص قيمتها في كل وقعة (دورة)من وقعات « ميقت » الوقت الفعلي ؛ وعندما تبلغ قيمة المتغير «صفرا» تُوضع المهمة في لائحة المهام الجاهزة مع الإشارة إلى الحالة الشاذة . ولزيادة فعالية هذه الأوالية ، تُرتَّب المهام حسب ترتيب التأخير المتزايد ، ومتغير التأخير يحتوي على وقت إضافي للانتظار لجهة المهمة السابقة في اللائحة . فقط متغير التأخير للعنصر الأول من اللائحة يجري إختباره في كل دورة لميقت الوقت الفعلي (timer realtime) .

وفي النهاية ، فإن المتغير « حالة » (statut) يحتوي على الحالة الراهنة للمهمة ويُستخدم متغيِّر الأولوية لترتيب المهام في لائحة المهام الجاهزة .

في نظام مثل UNIX ، يمكن للمهمة ، بواسطة الأمر «fork» ، إنشاء مهام «بنات» . الشجرة التراتبية للمهام سيتم حفظها بواسطة مؤشرات في واصف المهمة ، لأنه لا يمكننا القضاء على المهام من نوع « أقرباء parent » (أي رفع جميع اللوائح لجعلها غير موجودة) دون القضاء المحتمل على جميع المهام من نوع الأولاد (sons) «enfant» .

إضافة إلى جميع هذه المعلومات التي يستعلمها المرقاب ، يجب إضافة المعلومات المُستعملة في المستويات الأخرى من نظام التشغيل . مثلاً برنامج إدارة الإدخال ـ الاخراج يحتاج الى معرفة قنوات الإدخال ـ الإخراج (input-output channels) المُخصَّصة للمهمة ، برنامج إدارة الذاكرة يحتاج إلى مؤشرات نحو الصفحات أو القطع الخ . .

3.2 . طريقة عمل النواة

1.3.2 . إجراءات إدارة المهام

تتم إدارة المهام بواسطة نوعين من الإجراءات : إجراءات أساسية شفَّافة وإجراءات من مستوى أعلى ، مبلوغة للمستعمل . سنقوم بتفصيل هذه الإجراءات ؛ الأسماء المذكورة هي على سبيل المثال :

الاجراءات الأساسية تتعلَّق بـ :

ـ معالجة اللوائح .

INSERE : إدخال مهمة في اللائحة

INSEREH : إدخال مهمة في رأس اللائحة .

ENLEVE : إخراج مهمة من اللائحة .

AJOUTE : إضافة مهمة في طرف اللائحة .

- الترتيب والتبادل .

- ORDONNE (Shedule) : ويُحدِّد المهمة التي يجب أن تكون فعالـة في اللحظة المعينة وذلك بالبحث عن المهمة في رأس لائحة المهام الجاهزة .

- TRANSFERE (dispatch) : يؤدي إلى العبـور الفعلي إلى حـالـة المـرقـاب في التحكُّـم وذلك بتخزين نصّ المهمة التي كانت فعالة وبعـد ذلك بـاستخراج نصّ المهمة التي يجب أن تصبح فعّـالة ، وذلك بتعديل مضمون عداد البرامج .

- الاجـراء ORDONNE يعمل على لائحـة المهام الجـاهـزة وذلـك بـإعـادة تـرتيب اللائحة . يوجد خوارزميات مختلفة أكثر أو أقل تعقيداً جرى تطويرها . وتؤدي الى تداخل معايير مختلفة كمدة الوحدة المركزية المخصَّصة للمهـام ، تاريـخ الشحن والاطـلاق في الدوران والعمـل ، كمية الـذاكـرة المـطلوبـة لكـل مهمـة . بعض الخوارزميات تحاول إعادة ترتيب مجموعات المهام ذات الأولوية نفسها بينما غيرها يحاول تعديل أولوية المهام ديناميكياً قبل إعادة ترتيب لائحة المهام الجاهزة ، أما الإجـراء TRANSFERE فيحاول نقـل التحكّـم الى المهمة المـوجـودة في رأس لائحة المهام الجاهزة دون التعديل في هذه الأخيرة .

الإجـراءات ذات المستوى الأعلى هي مبلوغـة من المستعمل بـواسـطة بعض الأوامر .

هذه الأوامر هي عبارة عن إجراءات قابلة للتنفيذ بشكل غير قابل للإنقطاع (من هنـا إسم هذه الأوامـر) ، وتُطلب بـواسـطة مهـام المستعملين لتحـديـد التـزامن والاتصال . وتعمل بواسطة إجراءات أساسية . الأوامر الأكثر إستعمالاً هي :

- SIGNAL : إيقاظ مهمة في الانتظار لحادثة معينة .

- ATTENTE : يوضع في إنتظار إشارة معينة .

- SUSPENDRE : للتعليق (الوقف) .

- REVEIL : للايقاظ ، إيقاظ مهمة كانت معلَّـقة .

- P(S) : إختبار المُلوِّحة (Semaphore)

- V(S) : إطلاق المُلوِّحة (لوحة إشارات Semaphore) .

- ENVOI : إرسال رسالة (معلومات أو عنوان حيث توجد المعلومات) مباشرة إلى مهمة أخرى أو إلى علبة رسائل .

- RECEPTION : إستقبال رسالة (أو عنوان) بواسطة مهمة.الأوامر الستة الرئيسية الأولى هي عبارة عن أوامر للمزامنة . بينما الأوامر الأخيرة تسمح للمهام بالتزامن وتبادل المعلومات فيما بينها . الأوامر V وP تعود إلى مفهوم المِلْوحة «Semaphore» وهي مفيدة بشكل خاص في كل مرّة نرغب فيها بوصف القطاعات الحرجة ، أي أقسام البرنامج المنتمية لعدة مهام والتي لا يجب أبداً أن تُنفَّذ بشكل متزامن . هذا النوع من المسائل يظهر عندما نرغب بتقسيم الموارد المشتركة للحاسب بين عدة مستعملين . يتألف Semaphore من :

- عداد ،
- سجلّ انتظار ،
- عمليتين هما P وV .

العملية P تجري عندما ترغب إحدى المهام بالدخول إلى منطقة حرِجة ، وتنقّص من قيمة العداد « واحد » . وإذا أصبح مضمون العداد سلبياً ، فإن المهمة التي تقوم بهذا الاختبار توضع في الانتظار في « سجل الانتظار » المرتبط بـ Semaphore . العملية V ، التي تتم عندما نخرج من المنطقة الحرِجة ، تُزيد في العداد « واحداً » . إذا كانت القيمة الجديدة هي أيضاً سلبية أو صفراً عند ذلك فهناك على الأقل مهمة واحدة في الانتظار في السجل . واحدة منها يجري إيقاظها .

يوجد أيضاً أوامر تسمح بتنظيم الوقت وذلك بتعليق المهمة خلال عدد مُعيَّن من « دقّات » ميقت الوقت الفعلي أو بإدخال حراسة (chiens de garde) عند كل إنتظار . إذا كانت مدة إنتظار إحدى المهام (التي تنتظر إشارة أو رسالة) تزيد على القيمة القصوى فسيتم إيقاظها بواسطة مؤشر للخطأ .

وفي النهاية ، يوجد أوامر تسمح بتشكيل العمل التطبيقي وذلك بإنشاء مهام ، علب رسائل ، لوحات إشارة Semaphore ، أو بحجز مساحات عمل في الذاكرة الخ . . يتم إستعمالها خلال دوران العمل التطبيقي وتنفيذه .

2.3.2 . أوالية عمل المرقاب

أ ـ الشروط الداخلية

كما رأينا في الفقرة السابقة ، هذه الشروط تناسب بشكل أساسي تبادل

الاشارات (المزامنة) أو الرسائل (التبادل) بين المهام . سنقوم باختبار حالات تبادل الرسائل بين مهمتين بواسطة علبة رسائل في حالة إرسال العنوان مع تفادي نسخ الرسالة .

سنقوم باعتماد أمرين هما ENVOI(M, I) وRECEPTION (M, L) حيث M تُمثّل عنوان الحيِّز من الذاكرة حيث توجد الرسالة وL عبارة عن العنوان في لائحة الإنتظار المرتبطة بعلبة الرسائل المختارة . هذه اللائحة ستحتوي إما على عناوين الرسائل في الانتظار ، وإما على أسماء المهام (أو العناوين التي تؤشر نحو واصفات المهمة) المنتظرة للرسائل . ولن تحتوي أبداً على الإثنين معاً .

عندما تقوم المهمة Ti ، فعالة ، بتنفيذ RECEPTION (M, L) سيتم إستدعاء الاجراء المناسب للمرقاب الذي سيتم تنفيذه على الشكل التالي :

إذا (If) وُجِد رسالة في الانتظار في L
إذن (then) عودة إلى Ti مع العنوان M الموجود في رأس اللائحة L .
وإلا (else) إضافة (AJOUTE) Ti في طرف L ؛
إخراج (ENLEVE) Ti من لائحة المهام الجاهزة .
إضافة (AJOUTE) Ti إلى لائحة المهام في الإنتظار ؛
ترتيب (ORDONNE) لنفترض Tj المهمة المطلوب جعلها فعالة ؛
نقل (TRANSFERE) وإستخراج النص Tj . .
نهاية إذا (end of if) .

لنفترض الآن أن المهمة Ti فعّالة وتقوم بتنفيذ ENVOI (M, L) ، فالإجراء التالي سيتم تنفيذه .

إذا (if) L لا تحتوي على إسم المهمة .
إذن (then) M AJOUTE إلى L ،
عودة إلى Ti
وإلا (else) ENLEVE إخراج المهمة الواقعة في رأس L (أي Tj) ؛
إذا كانت Tj أكثر أولوية من Ti
إذن إدخال INSEREH Tj في رأس لائحة المهام الجاهزة ؛
تنظيم (ORDONNE) ؛
إرسال (TRANSFERE) ؛
وإلا (else) إدخال Tj (INSERE) في لائحة المهام الجاهزة ؛

عودة إلى Ti ؛
نهاية إذا (end of if)
نهاية إذا (end of if)

الإجراءات الأساسية المستعملة هي تلك المُعرَّفة في الفقرة 1.3.2 . من الممكن الإشارة إلى إستعمال ORDONNE قبل TRANSFERE التي تعيد ترتيب لائحة المهام الجاهزة في الحالة حيث الأولوية هي معدلة أوتوماتيكياً . من جهة أخرى ، من الممكن الملاحظة أن هذه الإجراءات الأساسية هي مستعملة على لوائح خاصة (اللائحة L في هذه الحالة) التي تأتي لتضاف إلى اللوائح الأساسية المعرَّفة في 1.2.2 . اللائحة العامة للمهام في الانتظار تعيد تجميع جميع هذه اللوائح وهي ضرورية لوضع **الحرس** (garde) في العمل المرتكز على إستعمال ميقت وقت فعلي .

ب ـ الشروط الخارجية

يضاف إلى كل إنقطاع :

ـ أولوية من العتاد يتم وضعها حسب المُعالِج المركزي المُستعْمَل .

ـ مهمة أو مجموعة من المهام في إنتظار الإنقطاع مع أولويتها الخاصة .

المهام المرتبطة بالانقطاعات هي دائماً في مستوى أولوية مرتفعة بالنسبة للمهام الأخرى بشكل يكون فيه النظام المعلوماتي مُتتبِّعاً لتطور العمل في محيطه بشكل سريع دون أية خسارة في المعلومات . ولكن على العكس يجب أن تكون المهام قصيرة الأمد لكي لا تقوم بإشباع المُعالج وشغْلِه بالكامل . لذا يؤدي ذلك إلى تقسيم المهام المرتبطة بالإنقطاع إلى قسمين . الحادثة المناسبة للإنقطاع تُعرَّف بواسطة مهمة قصيرة ، بأولوية كبرى يجري إيقاظها مباشرة بواسطة الإنقطاع . هذه المهمة يمكن أن تكون عبارة عن عملية كما هو معرَّف في 2.2 . في نهاية التنفيذ ، هذه المهمة تقوم بإرسال إشارة لمهمة معالجة بأولوية أقل وبمدة تنفيذ أطول .

وفي النتيجة ، يمكن القول إن عمليات المعالجة المتعلقة بالشروط الخارجية تُطلق في العمل بطريقتين :

ـ إما مباشرةٍ عندما تكون هذه المعالجة قصيرة بربط المهمة المناسبة بالإنقطاع العتادي .

ـ إما بشكل غير مباشر عندما يجري إطلاق المعالجة بواسطة مهمة عليها التعرُّف إلى الإنقطاع وتخزينه .

أخذ العلم بالإنقطاع وإطلاق المهمة المناسبة يمكن أن يتم بواسطة الإجراء التالي :

procedure TRAITEIT (LITi);

التعرُّف على الانقطاع

(منع الإنقطاع) ؛

مُخالصة الإنقطاع المعروف

قراءة لائحة المهام المرتبطة بهذا الإنقطاع (LITi) ؛

(لنفترض أن Ti هي المهمة الأولى من اللائحة) ORDONNE LITi

(إخراج Ti من LITi) ENLEVE Ti de LITi;

(إخراج Ti من لائحة المهام في الانتظار)

التأكيد على الإنقطاعات ذات الأولوية (عتاد) الأعلى من الانقطاع المتعرَّف عليه ؛

(إرسال التحكم إلى Ti) TRANSFERE le contrôle à Ti

نهاية الاجراء

يجب إجراء ملاحظتين هنا : الأولى تتعلّق باللائحة LITi وهي عبارة عن لائحة بالمهام التي تنتظر الإنقطاعات كاللائحة L التي تنتظر رسالة معينة . أي عندما يبدأ الإنقطاع فقط اللائحة Ti سيتم إيقاظها ، والمهام الأخرى تنتظر الإنقطاعات التالية . من جهة أخرى ، يجب ملاحظة أنه في النهاية يذهب التحكّم إلى Ti . هكذا ، فالمهام المرتبطة بإنقطاع غير ممنوع هي عادة أكبر أولوية من المهمة الفعالة ، لذلك كنا قد افترضنا إنها أصبحت فعالة .

الأجراء TRAITEIT كما هو معطي يفترض أن المهمة Ti كانت موضوعة في إنتظار الإنقطاع بتنفيذ الأمر ATTENDIT (LITi) التي يمكن أن توصف على الشكل التالي :

- إخراج Ti من لائحة المهام الجاهزة (Ti فعالة) ENLEVE Ti
- إدخال Ti في اللائحة LITi INSERE Ti
- إدخال Ti في لائحة المهام المنتظرة INSERE Ti
- (لنفترض Tj في رأس المهام الجاهزة) ORDONNE

- إرسال إلى Tj　　TRANSFERE TO Tj

ج ـ أخذ الوقت بالحسبان

ويتم بأخذنا كمرجع إشارة ميقت الوقت الفعلي . ويقوم :

- على تأخير المهمة خلال بعض الوقت (تخزين مؤقت) .
- إستخراج وإطلاق مهمة بشكل دوري .
- قطع إنتظار طويل لحادثة (حراسة) .

لنأخذ الحالة الأولى ، ولنفترض أن Ti هي المهمة الفعالة . ولنفترض أيضاً انها موضوعة في الانتظار حتى يمضي بعض الوقت قبل معاودة التنفيذ (مثلاً إعطاء النظام المحكوم الوقت الكافي للرد) . لهذا فمن الممكن إستعمال إجراء محدَّد على الطريقة التالية :

إجراء تأخير (قيمة)	**Procédure** RETARD (valeur);
إخراج Ti من لائحة المهام الجاهزة	ENLEVE Ti;
إدخال Ti في لائحة المهام المنتظرة بالحسبان	INSERETi;
القيمة المطلوب وضعها في متغيِّر التأخير للواصف	
نظِّم	ORDONNE;
إنقل .	TRANSFERE;

لنشر بشكل عام إلى أن مُتغيِّر التأخير يحتوي على المدة الإضافية التي يجب أن تمضي بعد إيقاظ المهمة السابقة في لائحة المهام المنتظرة . ساعة الوقت الفعلي هي عبارة عن إنقطاع بأولوية عالية يكون تنظيمها سهلاً على المُستعمل . ومعالجة هذا الانقطاع تختلف قليلاً عن المعالجة اللازمة لبقية الانقطاعات :

(لنفترض Tj المهمة الفعالة)　　procedure TRAITEHTR;

منع بقية الإنقطاعات ؛

تسوية الإنقطاع ؛

تنقيص عداد التأخير للمهمة الأولى من اللائحة المنتظرة «واحد» ؛

إذا كانت قيمة هذا العداد (المهمة Ti) هي « صفر »

إذن (then) إدخال Ti في لائحة المهام الجاهزة (INSERETi)

ترتيب؛　　ORDONNE

إذا Tj ليس في رأس اللائحة

إذن أنقل	TRANSFERE
وإلا عودة إلى	Tj
نهاية إذا	(end of if)

4.2 مثل على الإطلاق في العمل

سنبرهن ، في هذه الفقرة ، إنه من خلال موديل شكلي لشبكة بتري (petri) من الممكن بسهولة الوصول الى الإطلاق في العمل باستعمال أوامر نظام التشغيل في الوقت الفعلي . سنأخذ كمثل على ذلك نظام التشغيل iRMX/80 من شركة INTEL .

1.4.2 . أصول النظام iRMX/80

نظام التشغيل هذا هو أب النظام iRMX 86 الذي نذكر مبادئه العامة في القسم الثاني من هذا الكتاب . هنا لن نُقدِّم سوى أصول المزامنة والتبادل الضرورية لهدفنا . يتمّ التبادل بين المهام بواسطة الرسائل ويتمّ إرسال الرسائل بواسطة علب وسائل تُدعى «échanges» تحتوي إما على لائحة بالرسائل المنتظرة للاستعمال ، وإما على لائحة بالمهام المنتظرة للرسائل .

المبدأ RQWAIT يناسب المبدأ RECEPTION من الفقرة 2.3.2 والنحو هو :

MESSAGE : = RQWAIT (ECHANGE, TIMEOUT)

ECHANGE هو عبارة عن مُتغيِّر يدل على إسم علبة الرسائل حيث ستكون الرسالة موضوعة ، الاسم MESSAGE هو عبارة عن إسم المتحولة الداخلية من المهمة حيث سيتم نسخ الرسالة (فقط مؤشر الرسالة ، أي عنوانه ، سيتم نسخه) وTIMEOUT هو عبارة عن المدة القصوى للإنتظار (للتأخير) ؛ القيمة «0» تدل على أن الانتظار يمكن أن يكون غير محدود .

المبدأ RQSEND (يناسب ENVOI) ويتمتع بالنحو التالي :

RQSEND (ECHANGE , MESSAGE)

ECHANGE هو عبارة عن إسم علبة الرسائل وMESSAGE تحتوي على مؤشر

الـرسالـة . في الحالـة التي نرغب فيهـا بإرسـال إشـارة فقط دون إجـراء أي إتصـال معلوماتي ، يمكن أن نرسل رسالة فارغة موجودة في المتحولة MESSAGENIL .

وفي النهاية ، المبدأ الثالث جرى عرضه بواسـطة نظام التشغيـل iRMX/80 ، ويتعلّـق ذلك بِـ RQACPT مع النحو التالي :

MESSAGE : = RQACPT (ECHANGE)

إذا كان مؤشر الرسالة موجوداً في علبة الرسائل ECHANGE فسيتم نسخه في MESSAGE ، وإلا فإن رسالة فارغة سيتم إرسالهـا ، ولكن المهمة التي تُنجـز هذا المبدأ ليست أبداً في الإنتظار .

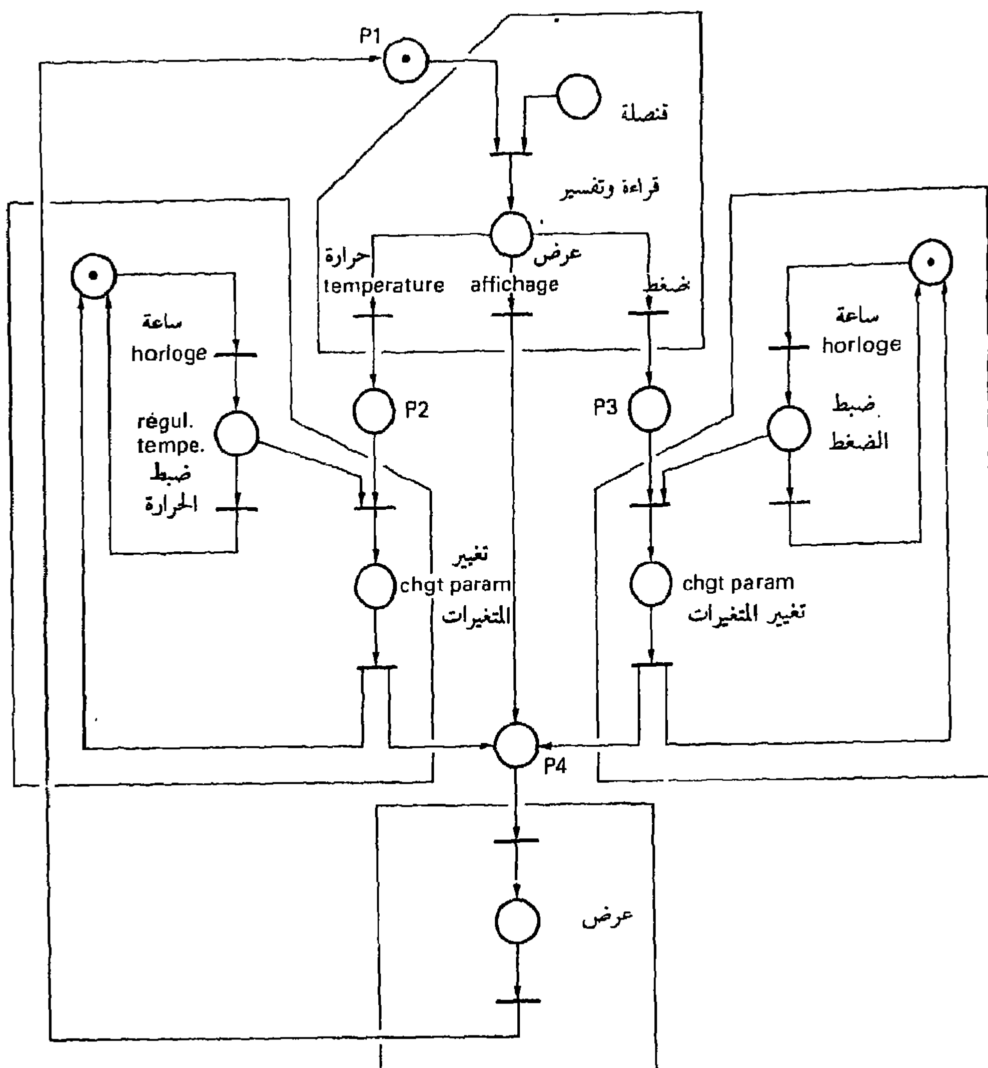
2.4.2 . البدء في التنفيذ من خلال النموذج الشكلي

لنستعـرض المثل من الفقرة 1.2 ونحدِّد الآن تعليق المهام مع المحـافظة على أكبر قدر ممكن من التوازي (parallelisme) . الشكل 3.2 يعطي شبكة بتري (petri) التي تصف التداخل بين المهام . تُمثل كـل مهمة في داخـل الشبكة بـواسطة مـوقع (على الأقل) . عندما يكون هذا الموقع مزوداً بإشارة ، فالمهمة هي في طور التنفيذ (في المستوى العملي) ؛ عندما لا يكون أي من المواقع المرتبطة بالمهمة مزوداً بأية علامة فالمهمة متوقفة (عملياتياً) .

عمليات الإنتقال للادخال والإخـراج للمواقـع المرتبـطة بالمهـام تناسب نقـاط التـزامن ، أي تلك التي تُمثِّـل إستعمال أصـول نظام التشغيـل . بشكـل عـام ، من الممكن إجراء التوازي بين المواقع غير المرتبطة بالمهام ومفهوم علبة الرسـائل . في الواقع ، تناسب الفيشة (jeton) المتنقّـلة بين هذه المواقع إشارة أو رسالة مُـرسلة من مهمة إلى أخرى بواسطة علبة الرسائل .

من جهة أخرى ، نفترض أن مهام الخدمة (البرامج المُساعدة) من نوع « تنظيم قنصلة المؤشر » ، تتصل مع غيرها بواسطة علب رسائل مُتخصِّصة (هـذه هي عامّـة الحالة مع iRMX/80) .

شبكة بتري في الشكل 3.2 تستخدم كمساعد للتقسيم بين المهام . لنفترض أن تعديل المتغيرات لا يمكن أن يكون متزامناً مع التنظيم ، من الممكن أن نقتـرح مثلاً تقسيماً بين الأربع مهام المشار إليها بالنقاط . نصل إذاً إلى بدء التنفيذ التالي بإعطاء ، حسب الإتفاق ، نفس الإسم إلى علب الرسائل والمواقع المناسبة .



شكل 3.2 : شبكة بتري (petri) لتعليق المهام

● مهمة القراءة والتأويل : MESSAGENIL : = RQWAIT(P1,0) ;

كرِّر بدون نهاية CHAR : = RQWAIT(CONSOLE,0) ;

● قراءة ، تأويل وإصدار العلامات CONSIGNE

●

في الحالة التي

« الحرارة » إذن — RQSEND(P2,CONSIGNE)

« الضغط » إذن — RQSEND(P3,CONSIGNE)

« طلب عرض » إذن — RQSEND(P4,COMMANDE)

نهاية الحالة

نهاية التكرار

مهمة العرض — Tâche d'affichage :

كرّر بدون نهاية — répéter sans fin

عرض مناسب للأمر — COMMANDE : = RQWAIT(P4,0)

-
- affichage correspondant à la commande
-

نهاية التكرار — RQSEND(P1,MESSAGENIL) ;
fin répéter

مهمة ضبط الحرارة وتعديل المتغيرات

Tâche régulation de température et changement paramètre :

كرّر بدون نهاية — répéter sans fin

ضبط

MESSAGENIL. : = RQWAIT(HORLOGE-TEMPERATURE,0) ;

-
- régulation
-

CONSIGNE : = RQACPT(P2) ;

إذا — si CONSIGNE < > MESSAGENIL

alors	إذن
•	
• changement de paramètre	تعديل المتغيِّر
•	نهاية إذا
fin répéter	نهاية التكرار

مهمـة الضبط وتعديـل الضغط هي متشـابهة مـع مهمة ضبط الحـرارة . يجب المـلاحظة أن التنفيـذ الدوري لهـذه المهام ، الـذي يتم بمدد دورات مستقلة ، هـو موضوع في العمل بواسطة علبتي الرسائل التي يرسل فيهما ميقت الوقت الفعلي رسائل بإيقاعات مختلفة .

3 . إدارة الذاكرة

تتألف ذاكرة الحاسب (تلك المبلوغة من المستعملين) من قسمين : الـذاكرة الحيَّـة (الـذاكرة المـركزيـة) ، المُشكَّـلة من دارات تكـامليـة (IC) ، والتي يبلغهـا الحـاسب بشكل سـريع ، ومن الـذاكرة الكبيـرة التي تستعمل نـواقل كـالأقـراص أو الأشرطة المغناطيسية والتي تكون مدّة بلوغها طويلة جداً .

الـذاكرة الحيَّـة هي عبـارة عن « مورد » حـرِج ومهم يعادل بـأهميــته أهميـة المعالج المركزي ، وإدارتها تكمن في جعل كل مُستعمل يتصوَّر بأنها تحت تصرفـه كاملة ، أو أنه يتمتع بذاكرة حيَّـة فرضية أكبر من الذاكرة الحقيقية . التقنية الأسـاسية تقوم على تخصيص حيزات (blocs) مختلفة من الذاكرة الحيَّـة لكل مستعمل (تقسيم المساحة من الذاكرة) وعندما تكون الذاكرة الفعلية غير كافيـة ، تقوم على تخصيص الذاكرة للمستعمل الذي يكون برنامجه في طور التنفيذ على المُعـالج ، بعـد أن يتم حفظ معطيات المستعملين الآخرين في الذاكرة الخارجية (نظام توزيع الوقت) .

الاستعمال الأفضل للذاكرة ينتج عن حلٍّ وسط بين المساحة من الذاكرة المُقدَّمة للمستعملين ومدة التنفيـذ التي قـد تـطول إذا كـان التخــزين (الحفظ) وإستخـراج (تـرميــم) المعطيـات يتم بشكل مُتكـرِّر بين الذاكـرة الحيَّـة والذاكـرة الخارجيـة . التقنيات المستعملة تستدعي عمليات أساسية سنقوم بذكرها سريعاً في المرَّة الأولى ، وسنعيد التذكير ببعضها بشكل مفصَّـل لاحقاً .

1.3 . العمليات الأساسية

تتألف إدارة الذاكرة من خمس عمليات أساسية هي :

- تحويل العنوان المنطقي إلى عنوان فيزيائي .
- تقسيم الذاكرة الفيزيائية عندما يُسمح لعدّة مهام باستعمال نفس القسم من الذاكرة .
- تخصيص الذاكرة الفيزيائية لعدة مستعملين مختلفين (توزيع المساحة من الـذاكرة وتوزيعها في الوقت) .
- حماية الحيـزات (الأقسام) المُخصَّصـة لكل مُسْتعمِـل (أو لنظام التشغيـل) ذات البلوغ غير المسموح به من جهة المُستعملين الآخرين .
- توسيع مساحة الذاكرة الى ما بعد الحـدِّ الفيزيـائي الذي يُشكّـل سعتهـا ، أي إلى الذاكرة الخارجية .

1.1.3 . تحويل العناوين المنطقية إلى عناوين فيزيائية

لتفادي الجمود في إستعمال الذاكرة ، فإن العناوين التي يعمل بهـا المُستعمل مباشرة هي عناوين منطقية (فرضية) وليست فيزيائية (فعليـة) . العملية تقـوم على التحويل من العناوين المنطقية إلى الفيزيائية والعكس (وتدعى هذه العملية memory mapping) ويمكن أن تتم في مختلف المـراحل حسب درجـة البساطـة والتعقيد في نظام التشغيل والبرامج المساعدة فيه .

يمكن تثبيت العناوين الفيزيائية عندما :

- يقوم المصرِّف (compiler) بتوليد الكود القابل للتنفيذ مباشرة .
- تُعالج الأقسام المصرَّفة بواسطة منقِّـح الأربطة (linker) لإنتاج الكود القابل للتنفيذ .
- البرنامج (الكود الراكن من جديد (relogeable) والذي يستعمل عناوين منطقية)هو مشحـون في الذاكـرة الحيَّـة من خلال الـذاكرة الخـارجية (loader) بغيـة تنفيـذه (ترجمة إلى عنوان ساكن) .
- البرنامج هو في طور التنفيذ (ترجمة ديناميكية) .

في الحـالة الأخيـرة ، ولأسباب نـاتجة عن زيـادة الفعـاليـة يتم إستعمـال دارة الكترونية متخصصة تدعى وحدة إدارة الذاكرة MMU (unit memory management). عملية وصف حيزات الذاكرة (segments) المُخصّصة من جديد للبرنامج تتم بشكل عام عند شحن البرنامج في الذاكرة الحية . هكذا دارة متخصِّصة موجودة مثلاً لدى المعالج MC68000 من شركة موتورولا .

2.1.3 . قسمة الذاكرة الفيزيائية

هذا الموضوع يفرض نفسه عندما تستعمل المهام تركيبة مشتركة للمعطيات . في الحالة الثانية يمكن أن نحلّ المسألة بجعل الزجلة مزدوجة ، وكل مستعمل يتمتع بنسخة في مساحة الذاكرة المخصَّصة له . عندما تكون الدارة MMU موجودة ، فإن الكود القابل للتنفيذ يمكن عدم نسخه في نسختين ، والدارة MMU تقوم بتحويل عناوين المعطيات لكل مستعمل في المساحات المناسبة .

عندما تكون تركيبة المعطيات مقسمة ، فإنّ نسخ المعطيات في نسختين يُصبح غير ممكن . يجب إذاً أن تكون المهام « مُزامنة » فيما بينها ، كي تبقى تركيبة المعطيات متماسكة . أصول المزامنة من نوع Semaphore يمكن أن تستعمل لادارة وتنظيم إستعمال هذه التركيبة بواسطة أقسام حرِجة .

3.1.3 . تخصيص الذاكرة

يتعلّق ذلك بتخصيص فدرات (بلوك) من الذاكرة الفيزيائية لمهام (برامج) مختلف المستعملين . هذا التخصيص يمكن أن يكون ساكناً بالكامل ، أي إن كل حيز من الذاكرة يجري تخصيصه للمستعمل مرة واحدة طوال مدة فعاليته (دورانه) .

ولا يمكن للمستعمل أن يزيد من مساحة الذاكرة المُخصَّصة له حتى ولو كان بقية المستعملين لا يستعملون كامل المساحة المخصَّصة لهم . ترجمة العناوين المنطقية الى عناوين فيزيائية تتم في لحظة شحن البرنامج (load).

عملية التخصيص يمكن أن تكون دينامية أي أن لا تخصص فدرة ذاكرة لمهمة معينة إلا عندما تحتاج لها هذه المهمة بشكل فعلي . ومن المحتمل ، طوال مدة تنفيذ المهمة أن يقوم نظام التشغيل بتحريك وإعادة تخصيص (dynamic relocation) فدرة مجهزة بعنوان فيزيائي إلى مهمة أخرى للحصول على إستعمال أفضل للذاكرة . هكذا ، فان المساحات من الذاكرة (الفدرات Blocs) لا تكون بشكل عام بحجم ثابت لذا فان كمية من الذاكرة بحجم مختلف يمكن عدم إستعمالها ليتم تخصيصها على شكل فدرات (blocs) . عملية التخصيص الديناميكي وتحريك الفدرات ليست ممكنة حتميا إلا في حالة وجود الدارة MMU وإلا فان مدة عملية تنظيم الذاكرة ستصبح طويلة .

4.1.3 الحماية

مسألة حماية الذاكرة تكون على وجهين :

- أن يكون النظام قادراً على اكتشاف عملية الدخول (عن قصد أو بدون قصد) الى مساحة من الذاكرة مُخصَّصة لمهمة معينة من قِبل مهمة غريبة ،

- أن يكون النظام قادرا على معالجة وإدراك عملية الدخول الى المساحة من الذاكرة المخصَّصة لمهمة معينة ، قبل أن يكون لها نتائج فعلية .

الحماية بواسطة وسائل منطقية هي عملية غير ممكنة من الناحية العملية بسبب مدة التنظيم والإدارة التي ستضاف الى مدة تنفيذ المهام . من جهة أخرى ، فإن التأكيد على صلاحية المهام قبل شحنها وتنفيذها (اكتشاف الأخطاء في البرمجة) ليس ممكنا . وعلى العكس عند وجود وحدة تنظيم الذاكرة MMU ، فان عملية الحماية يمكن أن تتم عند ترجمة العناوين المنطقية الى عناوين فيزيائية بشرط وجود جداول تخصيص للذاكرة تؤكد حقوق بلوغ المهام لفدرات (مساحات) الذاكرة المحددة لها .

5.1.3. توسيع مساحة الذاكرة

عدد البتات المُستعملة لتكويد العناوين في تعليمات الميكروبروسسور يُحدِّد المساحة القصوى للذاكرة الحيَّة (RAM) والتي لا يمكن أن تزداد دون إضافة أواليات خاصة . توسيع هذه المساحة يمكن أن يعالج فيزيائياً أو منطقياً .

ولاجراء توسيع فيزيائي للذاكرة ، يجب إستعمال وحدة إدارة للذاكرة MMU ، عندئذٍ تشكَّل الذاكرة في بنوك ، كل بنك يُناسب (يعادل) كامل المساحة المُعنونة . وعند ترجمة العنوان المنطقي إلى عنوان فيزيائي ، تقوم MMU بإضافة بتات قبل إرسال العنوان إلى وحدة التحكُّم (controller) بعمل وبلوغ الذاكرة . هذه البتات ، المُتناسبة مع رقم البنك هي محدَّدة عند تخصيص المهام بفدرات (بلوكات) للذاكرة .

عندما تكون الذاكرة الفعلية المُشكلة بواسطة مجموعة المساحة المعنونة (التوسيع الفيزيائي) غير كافية لاستيعاب العمل التطبيقي الذي نرغب بمعالجته ، فهناك عدة خيارات ممكنة :

- يعطي نظام التشغيل للمستعمل أصولاً إضافية تسمح له بتقطيع وتقسيم عمله التطبيقي .

- يضع نظام التشغيل في العمل أوالية خاصة للمستعمل كي يستطيع هذا الأخير العمل على ذاكرة فرضية أكبر من الذاكرة الفعلية .

تقسيم العمل المتناسب يمكن أن يتناسب مع التقسيم بين البرنامج الرئيسي ، وجميع المتغيرات العامة التي ستكون موجودة في الذاكرة الحيَّة، ومجموعة الإجراءات التي لن يتم شحنها إلا لتنفيذها (technique d'overlay) . الإمكانية الأخرى تكمن في تقسيم العمل التطبيقي في الوقت . فقط تركيبة المعطيات العامة هي دائماً موجودة في الذاكرة الحيّة ، أما قطع (أقسام) البرنامج فسيتم شحنها على التوالي بعد تنفيذ كلٍ منها (Technique de chaining) .

في الأنظمة من نوع مُتعدِّدة المهام، هناك تقنية يمكن إعتبارها « نصف - شفافة » وتقوم على تعليق مؤقت لبعض المهام وذلك بحفظ وتخزين كامل المساحة (المكان) من الذاكرة الحية المخصصة لها في الذاكرة الخارجية بغية إعادة تخصيص هذه المساحات لمهام أخرى . هذه الأوالية تدعى va et vient (Swap) وهي ليست عملية بالكامل ، لأنه في الحالة التي يكون فيها مستعمل واحد ، يجب تقسيم العمل التطبيقي إلى مجموعة من المهام المستقلة . من جهة أخرى ، فان تخزين وإعادة ترميم مساحات الذاكرة هو باهظ الثمن لجهة الوقت خصوصا عندما يكبر حجم المهام ، وهذه الطريقة هي غير ممكنة في الوقت الفعلي ، وليست قابلة للتطبيق دائما في نظام الوقت المقسم بين عدة مستعملين :

كي تكون المساحة المعنونة من الذاكرة المنطقية (عناوين منطقية) هي أعلى من المساحة الفيزيائية الموضوعة بتصرُّف المستعمل ، فإن نظام التشغيل يجب أن يقدِّم أوالية لادارة الذاكرة الفرضية . هذه الميكانيكية ترتكز على مفاهيم التصفيح والتقطيع (paging and segmentation) .

التصفيح (paging) يقوم على تقسيم الذاكرة الفيزيائية إلى فدرات (بلوكات) (page-frame) والذاكرة الفرضية إلى صفحات . الإرتباط بين الذاكرة الفيزيائية والذاكرة الفرضية هو ديناميكي وفي لحظة معينة لا نقوم بشحن سوى الصفحات التي نحتاج إليها في الذاكرة الحيَّة . أما الصفحات الأخرى فيتم تخزينها في الذاكرة الخارجية . هذا التقسيم إلى صفحات هو موضوع بتصرّف المستعمل ولا يوجد أي فرق بين البرنامج الرئيسي ، الإجراءات وإنشاءات المعطيات . وبهدف الحصول على حلول أفضل ، أي كي نتمكن من جمع وربط المعلومات المرتبطة فيما بينها في نفس الصفحات . نقوم بإدخال مفهوم القطعة (segment) . التقطيع (segmentation) يقوم بشكل خاص على فصل كود البرنامج عن المعطيات . هذا التقطيع يمكن أن يتم تحديده من قبل المُستعمل ولكن بإمكان المصرِّف أن يقوم به . من الواضح أن

عمليات التصفيح والتقطيع هي دون معنى إلا إذا كانت ترجمة العناوين المنطقية ـ الفيزيائية تتم ديناميكياً عند التنفيذ . وكنتيجة لذلك فإن ميكانيكية الذاكرة الفرضية ليست ممكنة إلا عندما تكون وحدة إدارة الذاكرة الفرضية (MMU) بتصرف المُعالج .

هذه الأوالية ، التي سنقوم بتفصيلها لاحقاً ، لا تزال مستعملة فقط على الحاسبات من الحجم الوسط (mini-computer) ، ولكن إستعمالها سيضم الميكروحاسبات بسبب صناعة وحدات تنظيم الذاكرة MMU مع الميكروبروسسور من نوع 16 و32 بتة .

2.3 . الذاكرة الفرضية

1.2.3 . التقطيع (Segmentation)

التخصيص الساكن للذاكرة هو ثابت (ويثبّت دائماً عند وضع النظام متعدِّد المستعملين في الخدمة مثل iRMX86 من إنتاج INTEL) ويؤدي إلى هدر في الذاكرة لأن المستعمل يمكن أن يرى أن مهمته غير منفَّذة بسبب الخسارة في الذاكرة بينما المساحة المخصَّصة لبقية المستعملين هي مستعملة .

التخصيص الديناميكي هو مُفضَّل ، ولكنه قد يؤدي إلى هدر في مساحة الذاكرة ناتج عن ظاهرة التفتيت أو التجزئة «fragmentation» . هكذا ، عندما ينتهي البرنامج ، فإن المساحة المخصصة له يجب أن تُسترجع لكي يتم تخصيصها بالتالي للبرامج الأخرى . إذا لم تكن الذاكرة مقسَّمة في فدرات بحجم ثابت ، وإذا لم يكن باستطاعتنا سوى تخصيص فدرة واحدة لكل برنامج كي تصبح المساحة من الذاكرة متكاملة ، فعندئذٍ ستصبح الذاكرة مُقطَّعة إلى قطعٍ صغيرة (fragments) صعبة الاستعمال بسبب صِغر حجمها وسيصبح من غير الممكن تنفيذ البرامج الكبيرة . إعادة إستعمال هذه القطع الصغيرة بجعلها متواصلة هو ممكن ولكنه صعب ، وأقل فعالية من التقنية المشروحة لاحقاً .

التقطيع يقوم على تقطيع المساحة المعنونة لأحد البرامج إلى قطع ليست بحاجة إلى أن تشغل مساحة متواصلة في الذاكرة . القطع تناسب التقسيم المنطقي لأحد البرامج إلى إجراءات، قطع ، برامج ـ ثانوية ، معطيات عامة أو مركزية محدَّدة بواسطة المستعمل .

الشكل 4.2 يعطي مثلاً على توزيع المهام في الذاكرة حسب مختلف القطع . القطع الثلاث الأولى تتعلَّق بنظام التشغيل : الأولى هي عبارة عن القسم الراكن من

النظام ، والثانية تحتوي على الجداول والمؤشرات التي تولّدها . هاتان القطعتان هما راكنتان (résidentes) أي سيتم شحنهما عند وضعهما في الخدمة وتخصيصهما لا يتعدَّل لاحقاً . القطعة الثالثة تستقبل أقساماً من نظام التشغيل مشحونة عند الحاجة من خلال الذاكرة الخارجية .

	رقم القطعة
نظام التشغيل الراكن	0
جداول نظام التشغيل	1
نظام التشغيل ، قطعة العمليات الخاصة رقم i	2
قطعة إجراءات الإدخال / الإخراج للمهمة 1	3
قطعة برنامج المهمة 2	4
قطعة المعطيات المقسَّمة بين المهمتين 1 و2	5
قطعة المعطيات الخاصة بالمهمة 1	6
قطعة البرنامج الرئيسي للمهمة 1	7

شكل 2.4 : تقطيع الذاكرة

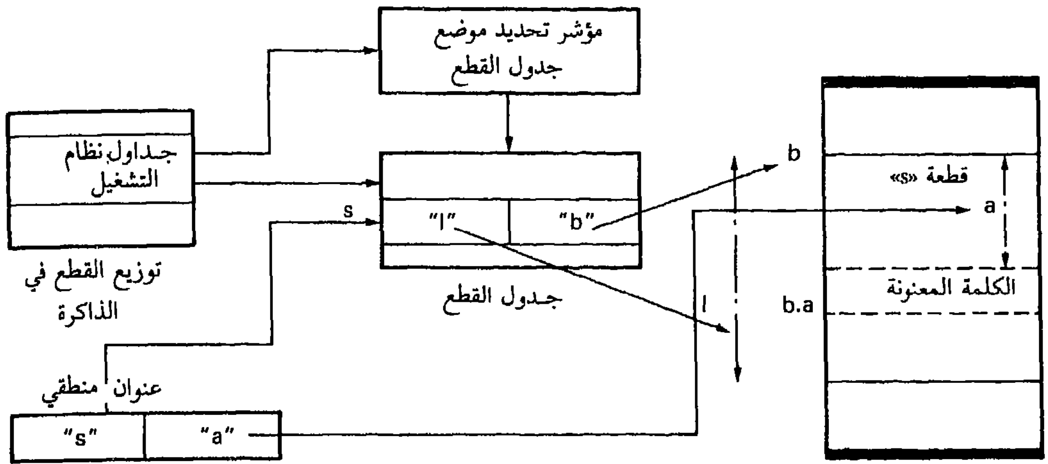
ضمن الجداول الموجودة في القطعة رقم 1 يوجد جداول للقطع (جدول لكل مهمة) . باستطاعتنا بلوغ جدول معين بواسطة مؤشر موجود في واصف المهمة المناسبة . أوالية العنونة تقوم على تقسيم العنوان إلى قسمين : القسم الأول يحتوي على رقم القطعة «s» ، والقسم الثاني يحتوي على العنوان النسبي «a» في القطعة . رقم القطعة «s» يسمح ببلوغ المتغيرات الوسيطة ، التي تصف وتحدِّد حالة القطعة في الذاكرة ، من جدول القطع (نفترض بأن ذلك يتعلَّق بقطعة مشحونة في الذاكرة الحيَّة) : طول القطعة «l» (طول القطع هو بشكل عام عبارة عن متغيِّر متحول) والعنوان الأساسي «b» . الشكل 5.2 يُوجز أوالية العنونة .

بالمقارنة بين «a» و«l» يمكن أن نحصل على الحماية (تفادي الدخول إلى مساحة من الذاكرة لقطعة مجاورة) .

إذا «a» ≥ «l» فإن العنوان هو صحيح

إذا «a» < «l» فهناك تداخل

تتم هذه العملية مباشرة بواسطة العتاد (دارة خاصة MMU) .



شكل 5.2 : أوالية عنونة الذاكرة المقطّعة

لنفترض أن القطعة هي عبارة عن وحدة ناتجة عن عملية تقطيع منطقية ، يمكن للقطعة أن تكون مُقطّعة بين عدة مهام (قطعة 5 من الشكل 4.2) . هذه القطعة سيتم تحديدها في أحد جداول قطع المهام المعتمدة . هذه التحديدات يمكن أن تكون مختلفة ، وبشكل خاص ، فإن حقوق البلوغ للقطع يمكن أن تكون مختلفة لكل واحدة من المهام (قراءة فقط أو قراءة ـ كتابة) .

يُستعمل التقطيع في بعض الأحيان في حالة التخصيص الساكن للذاكرة ، وبشكل خاص في أنظمة المعلوماتية المُخصّصة لعملٍ تطبيقي معيّن . وفي هذه الحالة يكون هدفها هو السماح باستغلال الذاكرة الفيزيائية المُشكّلة من فدرات بطبيعة مختلفة (ذاكرة ثابتة ، ذاكرة دائمة) . مثلا ، القطعة المناسبة للكود القابل للتنفيذ (البرنامج المستهدف) ستكون موجودة في الذاكرة الثابتة كي تتم حمايتها بالكامل ، بينما القطعة المناسبة للمعطيات سيتم تحديدها في الذاكرة الديناميكية.. في هذه الحالة ، فإن حساب العناوين الفيزيائية يتم عند الشحن (تشكيل النظام) ووحدة تنظيم الذاكرة MMU هي غير مفيدة .

حجم القطع هو متحوِّل ويتعلّق بشكل كبير بالتقطيع المعتمد من المبرمج . وبالنتيجة ، فإن ظاهرة التقطيع يمكن أن تتداخل بشكل واضح إذا كان تخصيص القطع في الذاكرة ديناميكياً . ولحلّ هذه الظاهرة يُستعمل مفهوم التصفيح (paging) .

3 .2.2. التصفيح (paging)

في عملية التصفيح يجري تقطيع البرامج والذاكرة الفيزيائية في آنٍ واحد .

- تُقسَّم الذاكرة الفيزيائية إلى فدرات (page frame)
- تقسَّم البرامج إلى صفحات .

بُعد الصفحات هو ثابت ويستجيب لاعتبارات خاصة بتنظيم العتاد (بشكل عام فان حجم الصفحات هو 1K ، 2K ، 4K) . وعلى قدر ما يكون حجم الصفحات صغيرا فان ظاهرة fragmentation التجزئة هي ضعيفة ، ولكن تنظيم هذه الصفحات يصبح أكثر تعقيدا ومدة هذا التنظيم تطول أكثر خارج كون بعد الصفحات ثابتا بينما حجم القطع هو متحول ، فان ميكانيكية العنونة هي متشابهة في التصفيح والتقطيع . هكذا فالعنوان ينقسم إلى قسمين :

- رقم الصفحة «j» (البتات ذات الأوزان العليا) .
- عنوان نسبي أو التحريك (depl) بداخل الصفحة «i» (البتات ذات الأوزان الضعيفة) .

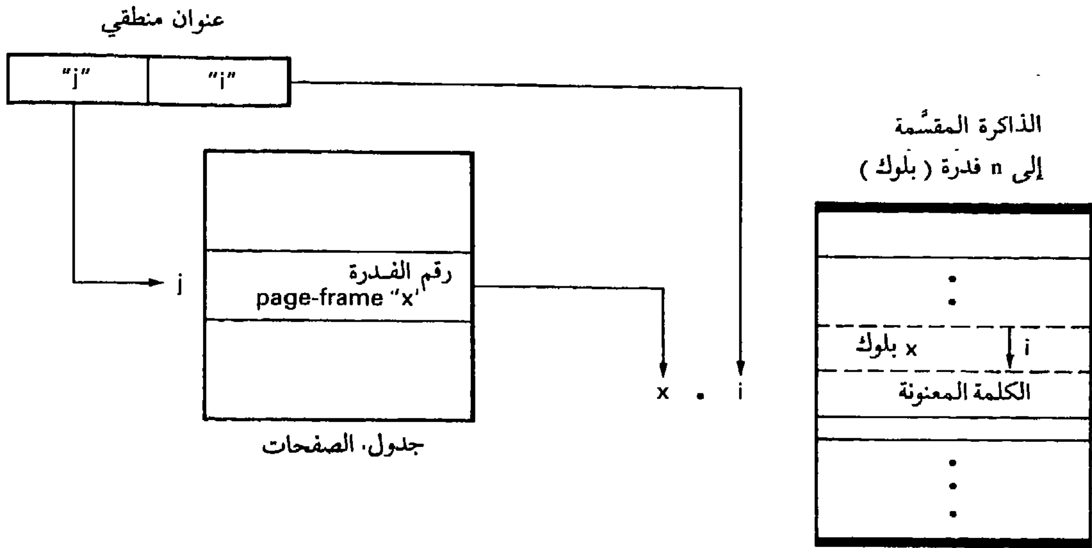
الترجمة الديناميكية للعنوان تستعمل جدولاً للصفحات حيث الموقع يتحدَّد إما بواسطة مِرصف (register) لتحديد موقع جدول الصفحات ، وإما بواسطة مؤشّر لجدول الصفحات موجود في واصف المهمة (أنظر الشكل 6.2) .

عندما يكون عدد الفدرات (p¬ge-grame) أقل بكثير من عدد الصفحات المستعملة بواسطة نظام التشغيل وبرامج المستعملين (وهذا ما يتناسب مع الاستعمال الطبيعي) فقد يحدث عادة أن يتناسب العنوان مع صفحة غير موجودة في فدرة (بلوك) معينة . وهذا ما يُسبِّب إنقطاعاً في العمل (فقدان الصفحة) (page fault) ، عندئذٍ يذهب نظام التشغيل للبحث عن الفدرة (page-frame) الموضوعة بتصرفنا للبحث بداخلها عن الصفحة المطلوبة ، لذلك يجب إستشارة جدول الفدرات (table of blocs) الذي يُزوِّد كل فدرة بالمعلومات التالية :

- حالة الفدرة (مشغولة أو حرة)،
- رقم الصفحة المشحونة بداخل الفدرة،
- مُعرِّف المهمة التي تنتمي إليها الصفحة .

وإذا لم يكن هناك صفحة حرَّة ، فيجب إختيار عملية تحرير واحدة وهذا

الإختيار يمكن أن يتم باستدعاء خوارزميات معينـة. ومن الممكن تحرير الصفحة التي يكون آخر إستعمال لها هو الأقدم ، أو تلك الأقل إستعمالاً .



شكل 6.2 : أوالية التصفيح

المسألة معقدة لأنه عندما يكـون عدد الصفحـات أكبر من عـدد الفدرات فـإن إحتمال ظهور عملية فقدان الصفحات هو كبير ، وهذا ما يؤدي إلى إنقطاع في معالجة المهام الجاري تنفيـذها ويـزيد بشكـل كبير من مـدّة إدارة الذاكـرة (نسخ الفـدرات المحرَّرة في الذاكرة الخارجية في المقابـل ، نسخ الصفحـة المطلوبـة هما عمليتـان باهظتا الثمن) . هذا ما يؤدي إلى حالة الإنتفاخ (thrashing)عندما تكون مدّة تنـظيم الذاكرة أكبر من مدّة الحساب المفيد . . لتفادي ذلك ، ولإجراء تقييم مبدئي من أي صفحـة يجب أن نقرأ المهمـة في الـذاكـرة جـرى تحـديـد مفهـوم مسـاحـة العمـل (working-set) : هو عبارة عن أصغر مجموعة من الصفحات التي يجب أن تركن في الذاكرة الحيّـة لتؤمن لأحد البرامج الفعالية المطلوبة فيما يتعلّـق بثمن تنفيذه . إذا كان عدد الفدرات الموضوعة بتصرف البرنامج أو المستعمل غير كافٍ لشحن مساحة العمل لمهمة معينة ، يجري تعليق عمل هذه الأخيرة بشكل مؤقّـت .

3.2.3 . التصفيح والتقطيع المتزامن

التصفيـح والتقطيـع هما عبـارة عن عمليتين بمبدأ واحـد ولكن بطريقـة عمـل مختلفة :

- مع التقطيع فإن تقطيع الذاكرة هو منطقي ولا يقوم به المستعمل بينما في التصفيح فإن تقطيع الذاكرة هو فيزيائي وبإرادة المستعمل .

- بُعد القطع هو متحول ويرتبط بطبيعة البرامج بينما تتمتع الصفحات والفدرات بأبعاد مثبتة بواسطة العتاد الذي يُدير عملية التكويد لبلوغ الذاكرة .

ربط هاتين التقنيتين يقوم على تقسيم البرامج إلى قطع بإشراف المبرمجين بينما يُدير نظام التشغيل عملية تعديل وتغيير الصفحات كي نحصل على أقل عدد ممكن من القطع الصغيرة جداً . تتألف العناوين عند ذلك من ثلاثة أقسام : رقم القطعة ، رقم الصفحة والتحريك (déplacement) .

تتمّ عملية ترجمة العنوان المنطقي إلى عنوان فيزيائي على الشكل التالي :

- رقم القطعة يشير إلى عنوان جدول الصفحات المرتبطة بهذه القطعة .
- رقم الصفحة يسمح ، في جدول الصفحات ، ببلوغ العنوان الأساسي (البتات ذات الأوزان العالية من العنوان الفيزيائي) .

- العنوان الكامل نحصل عليه بإضافة التحريك (déplacement) (البتات ذات الأوزان الضعيفة) إلى العنوان الأساسي .

الشكل 7.2 يُوجز هذه العمليات .

من البديهي أن يكون تتابع هذه العمليات لحساب العنوان الفيزيائي لكل بلوغ بسيط للذاكرة الحيَّة غير ممكن إذا تم ذلك بواسطة المعالج المركزي بشكل كلاسيكي ، أي بواسطة المناهج فقط . وحدات تنظيم الذاكرة (MMU) تقوم بهذه العمليات ، وهي تستعمل تجهيزات من العتاد متخصِّصة بذلك وتستطيع طلب ذاكرات ترابطية لتخفيض مدّة البحث عن مختلف الجداول .

عنوان منطقي (فرضي)

قطعة | صفحة | تحريك

عنوان فيزيائي

+

رقم القطعة

رقم الصفحة

1

2

s

i

الصفحات عنوان جدول

جدول القطع

جدول صفحات القطعة 1

العنوان الحقيقي الأساسي

1

2

3

جدول صفحات القطعة s

1

2

جدول صفحات القطعة رقم i

شكل 7.2 : التقطيع والتصفيح مختلطين

الفصل الثالث

تنظيم عمليات الإدخال ـ الإخراج
input - output

1 . مدخل

يهدف تنظيم عمليات الإدخال ـ الإخراج إلى عزل المُستعمل عن مواصفات العتاد . تتصل المهام التطبيقية مع مداخل ـ مخارج منطقية بشكل نموذجي عام . وقد أصبح ذلك ممكناً بوجود منظم لعمليات ووحدات الإدخال ـ الإخراج (input-output manager: IOM) الذي يستخدم كملقى بين الرؤية المنطقية والفيزيائية للوحدات المحيطية من خلال مجموعة خدمات يستطيع المُستعمل بلوغها .

هناك طريقة لجعل الملقى بين مهام المُستعمل ومنظم الإدخال ـ الإخراج IOM نموذجياً ، وتكمن باعتبار هذه الأخيرة(عمليات الادخال ـ الإخراج) كسجلات خاصة تستطيع المهام أن تقرأ وتكتب فيها . مفهوم هكذا مُنظِّم يرتبط بالمستوى الأعلى لنظام التشغيل المتعلّق بتنظيم السجلات . لجهة العتاد فإن منظّمات الإدخال ـ الإخراج يجب أن تتصل بمنظّمات الوحدات المحيطية (device mangement) أو (device driver)التي تشتغل بعمليات التبادل بين المعطيات الفيزيائية على مختلف الأجهزة المحيطية : نعتبر ان هذه المنظّمات هي قسم من العتاد المحيطي أكثر من كونها قسماً من نظام تشغيل الحاسب .

ولقد رأينا في الفقرة الأولى ان المبدأ الأساسي لمنظّمات الإدخال ـ الإخراج هو في جعل الرؤية المنطقية مستقلة عن وحدات الإدخال ـ الإخراج . وبشكل خاص ، يجب أن تسمح لمهام المستعمل بالإتصال مع الوحدات بواسطة أسماء . منطقية (اسم سجل خاص مثلاً) ، مما يسمح بتعريف ديناميكي لوصلة المهمة مع

وحدة معينة خلال تنفيذها (كما يجري عندما نفتح سجلاً معيناً لتسجيل المعلومات بداخله) .

كي نستطيع أن نكون مستقلين عن وحدات الإدخال ـ الإخراج يجب أن لا نكون مرتبطين بسرعات الأدوات المحيطية في تبادل المعلومات . هكذا ، فسرعة الأدوات المحيطية هي ضعيفة عادة بالمقارنة مع إمكانيات تبادل المعطيات مع الوحدات المركزية ، ومن جهة أخرى فهي مختلفة فيما بينها . وكي لا يتم حصر مهام المستعمل بعمليات إدخال ـ إخراج نستعمل ذاكرة دارىء (Buffer) بإنشاء أوالية « للإنتاج ـ الاستهلاك » في العمل . عند الإخراج ، فإن الدارىء سيكون مملوءاً بضربة واحدة وسيتم إطلاق مهمة خاصة (يمكن أن يتم تنفيذها بواسطة مُعالج خاص) في العمل بشكل مُنفصل عن إجراء عمليات التبادل مع الوحدات . هذا التنظيم للدارىء ولأوالية « الانتاج ـ الاستهلاك » سيُصبح تحت إشراف المُستعمل بسبب وجود مُنظِّم الإدخال ـ الإخراج .

ولقد رأينا أعلاه أن الوسيلة الأسهل لجعل الملقى بين مهام المستعمل ومنظِّمات الإدخال ـ الإخراج نموذجياً كانت باعتبار هذه الأخيرة وكأنها عبارة عن سجلات (files) خاصة . وهذا يتطلب ان تكون المعطيات مركّبة في تسجيلات (records) نستطيع تبادلها مع مُنظِّم عمليات الإدخال الإخراج بواسطة قنوات (channels) . القنال هو عبارة عن الرؤيا المنطقية الديناميكية للمستعمل بالنسبة لوحدة الإدخال ـ الإخراج . مثلاً ، نفتح قنال الإخراج نحو السجل الخاص « بالطابعة » ونكتب فيه تسجيلات « أسطر من السمات الأبجعددية » . يؤلف القنال pipeline الوصلة بين المُهمة المُستعمِلة وسجلّ خاص يناسب الأداة الطرفية الفرضية (terminal virtuel) . يرتبط بكل قنال دارىء (buffer) واحد بحيث تقوم مهام المُستعمل بكتابة التسجيلات . القنوات هي مفتوحة ومغلقة ديناميكياً بواسطة مهام المستعملين خلال تنفيذها .

في مفهوم متعدِّد البرمجة تستطيع عدة مهام بلوغ نفس الأداة المحيطية ؛ البلوغ المتعدِّد يمكن أن يتم بإشراف وحدة إدارة خاصة . المُنظِّم IOM يراقب ويتحكّم بطلبات العمليات على هذه الأدوات الطرفية ويمكن أن يقوم بتأخير بعضها لتأمين بعض التزامن والتماسك في التنظيم : ولتفادي وضع مهام المُستعملين في « الانتظار » ، يمكن للمُنظِّم IOM أن يطلق في العمل تقنية خاصة تدعى «spooling» وتقوم على إستعمال ناقل خاص (بشكل عام اسطوانة) مثل « دارىء

خارجي » لتخزين الطلبات بشكل متكامل وتنفيذها .

2 . تركيبة المُنظِّم ، القنال

كما أشرنا في المدخل ، فإن مُنظِّـم الإدخال ـ الإخراج (IOM) يرتبط بعـلاقة من جهة مع مُنظِّم السجلات ومن جهة أخرى مع الوحدات الفيزيائية . وعندما يفتح أحد المستعملين سجلاً معيّناً أي عندما يُصرِّح عن رغبته باستعمال وحدة للإدخال ـ الإخراج (فتح سجل خاص) ، فعند ذلك يقوم مُنظِّم الإدخال ـ الإخراج بالتحكّم وإدارة جميع طلبات العمليات على السجلات (قراءة أو كتابة) .

1.2 . تركيبة المُنظِّـم (IOM)

من الممكن إعتبـار أن المُنظِّـم يتـألف من مُستويين ـ ثـانويين ، الأول يتعلّق بمعالجة المعطيات المنطقية (ويُدعى غالباً LIOCS : Logical Input Output Con- trol System) ، والآخر يتعلّق بمعالجـة المعطيـات الفيزيـائية (ويُـدعى PIOCS : Physical Input Output Control System) أو BIOS(Basic Input Output System)-

بين هذين المستويين يوجد ملقى يقوم بتحويل المعطيات المنطقية إلى معطيات فيزيائية وبالعكس .

أ ـ LIOCS

يسمح باعتبار المعطيـات مـوجـودة على شكل تسجيـلات منطقيـة (records) مُخـزَّنـة في سجـلات خـاصـة . هـذا المستـوى يُشـرف على عمليـة البلوغ لجهـة المستعملين بـاستعمـال ذاكـرات داريء (buffers) للحصـول على إستعمـال أفضـل للمعالج أي لجعل المهام موضوعة أقل قدر ممكن في « الإنتظار » عندما تقوم بإجراء طلبات للإدخال ـ الإخراج .

ب ـ BIOS

ويعمل على عناوين حقيقية . ويحتوي على زُجلِ تتحكّم باستراتيجية إستعمال الوحدات الفيزيائية ، مثلاً يمكن أن يجمع بين طلبات متعـدِّدة على نفس الوحـدة . ويتألف أيضاً من زُجل تحكّـم بعمليات الإدخـال ـ الإخراج (device handler) على علاقة مباشرة مع مُنظِّـم عمل الوحدات الفيزيائية (device driver) ومع مُنظِّـم عمليات الإنقطاع (interrupt manager) .

2.2 . مفهوم القنال

القنال هو الـوسيلة التي بواسـطتها يستـطيع البـرنامـج أن يبلغ سجلاً أو وحـدة

خارجية . عند فتح السجل ، يقوم المُنظِّـم بإضافة رقم قنال إلى استعمال السجل . بعد ذلك يذكر المستعمل هذا الرقم عند كل طلب للإدخال ـ الإخراج (في الأنـظمة الأكثر قدرة يمكن لهذا الرقم أن يُستبـدل بواسـطة قسم منطقي) . الفـائدة من هـذا المفهوم لرقم القنال هو في كونه مختلفاً في كل سلسلة من عمليات إستعمال وحدات الإدخال ـ الإخراج ويختلف حسب نوع عملية إستعمال الوحدة أيضاً : مثلاً ، سيكون للقنصلة إسم سجل خاص (Cons 10 مثلاً) ولكن قد يكون لها رقم للقنال «1» عند الإدخال والرقم «2» عند الإخراج .

تخصيص القنوات يمكن أن يكون عاماً أو مركزياً . في الحالة العامة ، فإن نظام التشغيـل يُعرِّف عـدداً ثابتـاً من القنوات عنـد توليـد النظام ووضعـه في العمل . من الممكن إذاً أن نلتقي إحدى الحالات التالية :

- يُعـاد تخصيص القنوات الى وحـدات الإدخال ـ الإخـراج عند التشكيـل ، والمهام التطبيقية يجب أن تطلب رقم قنال مناسباً لعمليـة الإدخال ـ الإخـراج التي نرغب بإجرائها ، وفي حالة عدم توفّـر هذه القنوات توضع المهام في الانتظار .
- تتطلب المهمة التطبيقية قنالاً حُرّاً لكي تتمكن من تخصيصـه إلى وحدة لـلإدخال ـ الإخراج ، وإذا لم يكن هناك أقنية توضع المهام في الانتظار .

في الحالة المركزية ، كل مُستعمل يديـر مجموعـة قنواتـه الخاصـة به . هـذه الحالة ، الأكثر مرونة ، هي صعبة للتنظيم والإدارة لأنها تسمح لعدة مستعملين بإجراء عمليات الإدخال ـ الإخراج على نفس الوحدة .

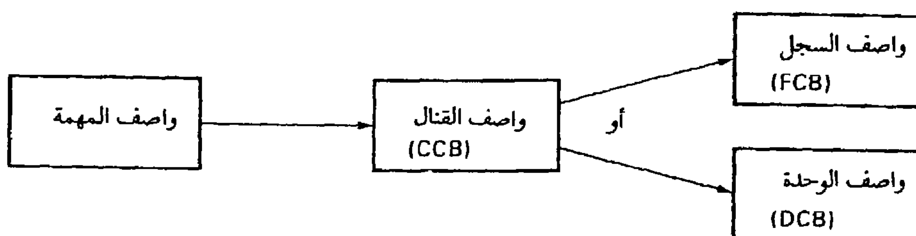
يُضاف إلى كل قنال واصف للقنال (channel descriptor) أو channel control block: CCB الذي يحتوي على جميع المعلومات عن الحالة الجارية للقنال . وعندما يرتبط أحد القنوات بسجل معين ، يجري إعداد الواصف بالعناصر المميِّـزة للسجـل (أو لوحدة الإدخال ـ الإخراج) واستعماله . مثلاً ، من الممكن أن نجد المعلومات التالية :

- تعريف طريقة البلوغ (القراءة أو الكتابة) .
- مؤشر نحو واصف السجل FCB (file control block) (أو الوحدة Device control block: DCB) .

يوجد هذا الواصف في المساحة من الذاكرة المحفـوظة للنـظام . وعندمـا يتم تخصيص أحد القنوات بمهمة تطبيقية معينة فإن واصف القنال يرتبط بالمهمة بواسطة

مؤشر موجود في واصف المهمة (أنظر الفصل التالي) ، كما هو مشار إليه في الشكل 1.3 .

يجب تحديد عدد القنوات الجاهزة (المحدَّدة عند تشكيل النظام) ومن الضروري ربطه بالنظام (عدد المستعملين بالتزامن مع عدد وحدات الإدخال ـ الإخراج) . بما أنه يرتبط بكل قنال واصف للقنال حتى ولو لم يتم تخصيص هذا القنال ، لذا فتعريف عددٍ كبير من القنوات يؤدي إلى خسارة في الذاكرة . وفقط في أنظمة التشغيل بالوقت الفعلي المستعملة للتحكّم بالعمليات الصناعية ، يمكننا تحديد دقيق لعدد القنوات المطلوبة والتي يجب إنشاؤها عند تشكيل نظام التشغيل .



شكل 1.3 : ربط واصفات السجلات والوحدات في عملية للإدخال ـ الإخراج

3 . الداريء (Buffer)

مهمة الداريء هي مزدوجة :

- يُستخدم كمُكيِّف (adapter) بين الوحدة المركزية ووحدات الإدخال ـ الإخراج التي تعمل بسرعات مختلفة ، والسماح باستعمال أفضل للمعالج .
- حلّ مسألة الاختلاف في حجم وتركيبة التسجيلات المنطقية (المرئية من المُستعمل) والتسجيلات الفيزيائية (المرئية لوحدات الإدخال ـ الإخراج) .

1.3 . مبدأ عمل الداريء

عندما يكون القنال مخصصاً لإحدى المهام ، فسيتم تخصيص مجموعة من الأحواض (Pool) من ذاكرات الداريء الضرورية لها . إستعمال هذه الذاكرات هو ديناميكي .

مثلاً ، عندما يتم تخصيص القنال إلى مهمة في القراءة ، فسيتم تسجيل التسجيلة الأولى (بواسطة BIOS) في الداريء الأول . المهمة المُستعملة تقرأ التسجيلات المنطقية في هذا الداريء (بواسطة LIOCS) ، وعندما يتم إستهلاك

كـامل مضـمـون الداريء ، يتم تخصيص داريء جـديد مملوء إلى القنـال (بواسـطة BIOS) .

عندما يكون القنال مُخصَّصاً للكتابـة ، فسيتم تخصيص داريء فارغ للقنـال . المهمة المُستعملة ستكتب فيه التسجيـلات المنطقيـة (بواسـطة LIOCS) . عندمـا يكون الداريء مملوءاً ، فسيتم نقله الى وحدة الإخراج على شكل تسجيلات فيزيائية (بواسطة BIOS) ويتم تخصيص داريء جديد للقنال .

يجـري إسترجـاع ذاكرات الـداريء المستعملة في الأحـواض (Pool) لإعـادة تخصيصها الى القنال . هكـذا ، فتنظيم الـداريء هذا يمكن أن يكـون أكثر أو أقـل تعقيداً وسنقوم بتفصيل المفاهيم الرئيسية التي نلتقيها في الفقرة التالية .

2.3 . تخصيص وتنظيم الداريء

يمكن أن نلتقي ثلاث مهام رئيسية :

- مجموعة من الأحواض (pool) من ذاكرات الداريء .
- الداريء الدائري .
- الداريء المزدوج .

1.2.3 . مجموعة من أحواض ذاكرات الداريء (pool buffers)

هذا هو مبدأ التقنية الأساسية الذي عرضناه في الفقرة السابقة . عند تخصيص القنال إلى مهمة معينة ، يتمّ حفظ مجموعة ذاكرات الداريء في المساحة الفارغة من الذاكرة . يتمتـع كـل داريء بـأبعاد تتنـاسب مع حجم التسجيلة الفيـزيائيـة (عدد مُضاعف لـ 128 بايتة مثلاً) . إلى كل داريء تجري إضافة واصف (Buffer Control Block: DCB) .

يرتبط كل واصف بالآخر بواسطة مؤشرات ويحتوي عادة على العناصر التالية :

- مؤشر نحو الداريء (pointer) ،
- الأبعاد المنطقية للداريء (يمكن أن تكون مختلفة عن الأبعاد الفيـزيائيـة التي هي عبارة عن عدد مضاعف لقدرة الذاكرة) ،
- دليل (index) يعطي الصفحة الفعلية من الذاكرة التي تحتوي على الداريء ،
- حالة الداريء .

هذه الحالـة تشير إلى مـا إذا كان الـداريء فارغـاً (جاهـزاً لكتابـة) أو ممتلئاً

(جاهزاً للقراءة) . عملية تخصيص ذاكرات الداريء لإجراءات القراءة والكتابة الموجودة في البرامج LIOCS و BIOS تتم على أساس الداريء الأول الجاهز . الفائدة من هذه التقنية تكمن في عدم كون عدد ذاكرات الداريء مثبتاً من البداية . وعندما يكون القنال مزدوجاً (كتابة وقراءة متزامنة على نفس الوحدة) ، فإن مجموعة ذاكرات الداريء Pool ، يمكنها ، مع بعض الإحتياطات ، أن تكون مشتركة بين القناتين .

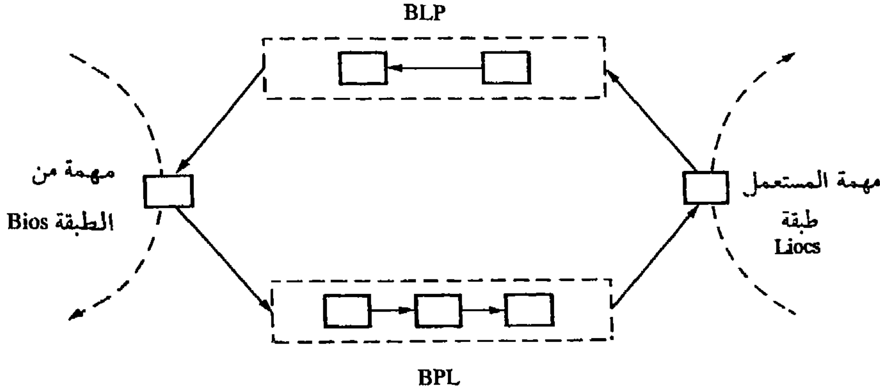
تنظيم ذاكرات الداريء يمكن أن يتم بواسطة «علبتين من الرسائل» . عند فتح القنال ، يجري إنشاء علبتي الرسائل ، الأولى وتدعى BLP تستخدم للتبادل في المستوى المنطقي نحو المستوى الفيزيائي ، والأخرى وتدعى BPL، تستخدم للتبادل من المستوى الفيزيائي باتجاه المستوى المنطقي (أنظر الشكل 2.3) . نحصل إذاً على طريقتي عمل مختلفتين بشكل ضعيف فيما بينهما حسبما يكون القنال للقراءة أو للكتابة .

إذا كان القنال هو قنال للقراءة ، فإن واصفات (descriptors) ذاكرات الداريء (BCB) تكون مرتبطة بالواصف المركزي في العلبة BLP لأن هذه الأخيرة تناسب لائحةذاكرات الداريء الموضوعة بالتصرف بالنسبة للمهمة BIOS التي يقع على عاتقها مهمة نسخها . هذه الأخيرة تقوم بتعبئة جميع ذاكرات الداريء الجاهزة والموجودة في BLP وتقوم بإرسالها في BPL ،وعندما تكون BLP فارغة ، وتوضع بعد ذلك في الانتظار . خلال هذا الوقت ، فإن مهمة المُستعمل تقوم بتنفيذ إجراء (procedure) من طبقة LIOCS في كل مرّة نلتقي فيها أمراً بالقراءة . هذا الإجراء يقوم على أخذ الداريء الأقدم الجاهز من علبة الرسائل BPL وقراءة المعلومات منه والموجودة على شكل تسجيلات منطقية .

العلبة BPL تحتوي إذاً على واصفات ذاكرات الداريء المُعبّأة بواسطة BIOS والجاهزة للاستهلاك لكي يتم إستهلاكها بواسطة LIOCS . وإذا كانت هذه العلبة فارغة توضع مهمة المستعمل في الانتظار . عندما تكون جميع المعلومات الموجودة في الداريء قد جرى إستهلاكها تقوم مهمة المستعمل بإرسالها إلى العلبة BLP .

هذه الأوالية مُخطّطة على الشكل 2.3 حيث جرى تصوير ذاكرات الداريء بواسطة مستطيلات . هذا الشكل يُحدِّد حالة pool من 7 ذاكرات داريء ثلاث منها جرى تعبئتها بواسطة BIOS وتنتظر إستهلاكها من قِبل LIOCS ، واثنتان منها

فارغتان . البرنامج LIOCS هو في طور إستغلال داريء وبالتزامن مع ذلك يقوم BIOS بتعبئة داريء آخر .

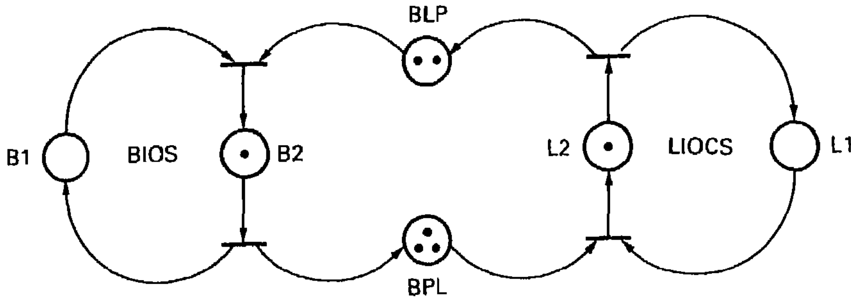


شكل 2.3 : تنظيم pool من ذاكرات الداريء

جرى تحديد التشغيل الديناميكي بواسطة شبكة بتري (petri) على الشكل 3.3 حيث تمّ تصوير واصفات الداريء بواسطة فيش . الموقعان BLP وBPL يمثلان علبتي الرسائل من نفس الإسم . الموقع L2 يُمثِّل مهمة المستعمل في طور قراءة الداريء بينما الموقع L1 يمثل مهمة المُستعمل في طور إنتظار داريء أو في طور إجراء عمل آخر . بشكل متشابه ، فإن الموقع B1 يُمثل دارئاً بمضمون تسجيلة فيزيائية . شبكة بتري تمثل التعاون المشترك بين الطبقات LIOCS وBIOS الناتج عن تنظيم القنال بواسطة مجموعة من ذاكرات الداريء .

في حالة قنال الكتابة ، عند الفتح يجري إرسال جميع واصفات ذاكرات الداريء في العلبةBPL التي تحتوي على واصفات الداريء الفارغة . تقوم مهمّة المستعمل باستهلاكها خلال المدة التي تقوم فيها بتنفيذ أوامر الكتابة وتقوم بارسالها الى العلبة BLP التي تحتوي على « واصفات » ذاكرات الداريء المكتوبة بواسطة LIOCS والجاهزة لكي يتم نسخها على ناقل فيزيائي بواسطة BIOS .

الإدارة الديناميكية لذاكرات الداريء هي شبيهة بالحالة السابقة (أنظر الأشكال 2.3 و3.3) . هنا نرى أنه في نظام متعدِّد المهام تكون مهام البرامج LIOCS وBIOS هي متزاوجة كي تكون جاهزة للعمل بالتزامن (هذه هي الحالة في الشكلين 2.3

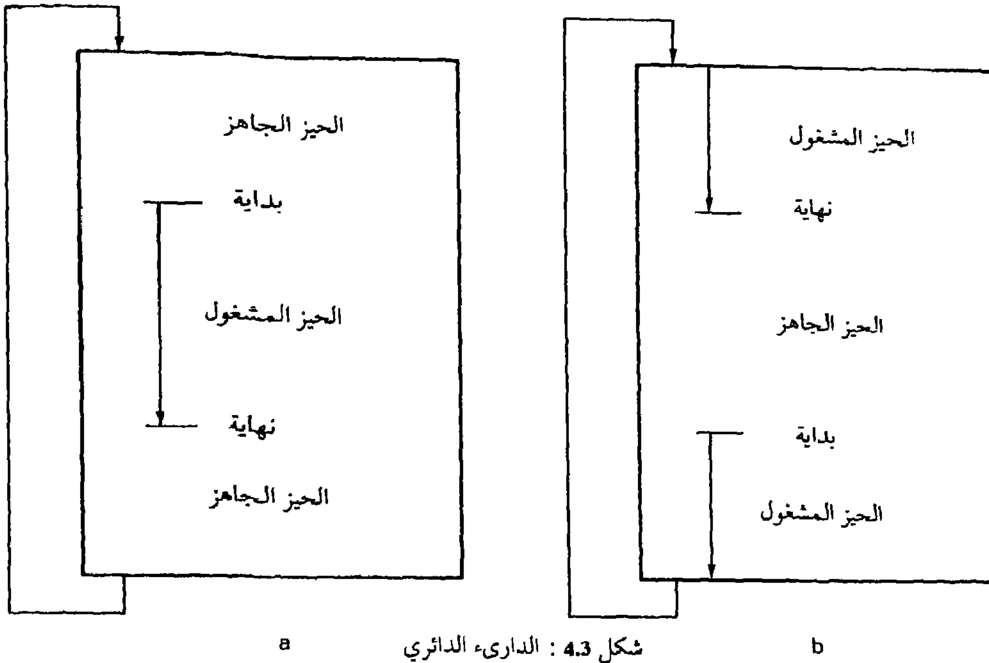


شكل 3.3 : التنظيم الديناميكي لمجموعة ذاكرات الداريء (pool)

2.2.3 . ـ الداريء الدائري

يناسب الداريء الدائري حيزاً من الذاكرة يعادل مُضاعف بعد أو حجم تسجيلة منطقية . نبلغ هذا الداريء بواسطة مؤشّرين « بداية » و« نهاية » . لنأخذ مثلاً حالة داريء كهذا مُستعمل عند الإدخال . ففي الحالة الأولية ، « بداية » و« نهاية » تؤشران نحو نفس العنوان ، وBIOS يكتب التسجيلات المنطقية من خلال « نهاية » ويعيد تركيز المؤشّر على نهاية آخر تسجيلة فيزيائية مكتوبة . وقبل كل عملية كتابة يتحقق من أن « نهاية » لن تقوم باجتياز « بداية » . LIOCS ، نفسه ، يقرأ التسجيلات المنطقية من خلال « بداية » بعد أن يكون قد تحقّق من كون هذه التسجيلات هي كاملة أي أنها لا تزيد أبداً عن « نهاية » . بعد كل عملية قراءة تُركّز « بداية » على بداية التسجيلة الأولى المنطقية للقراءة . بين « بداية » و« نهاية » نجد إذاً المعلومات الناتجة عن عمل BIOS وغير المستعملة بواسطة LIOCS ، من « نهاية » باتجاه « بداية » نجد المساحة الجاهزة من الذاكرة حيث يستطيع BIOS الكتابة .

هذه الأوالية تشبه مجموعة ذاكرات الداريء (pool) ولكن بطريقة معينة فإن LIOCS يرى ذاكرات داريء حيث أبعادها عبارة عن تسجيلات منطقية (سطر مثلاً) بينما BIOS يرى ذاكرات الداريء حيث الحجم المناسب للتسجيلات الفيزيائية (بلوكات من 128 بايتة مثلاً) . الشكل 4.3 يظهر لنا بدء العمل بداريء دائري في حيّز متواصل من الذاكرة . في a.4.3 الحيّز المشغول ، أي الذي يحتوي على معلومات منتجة بواسطة BIOS ، والتي ليست مستهلكة ، هي متواصلة بينما في b.4.3، من الناحية المنطقية لن يكون قد تغيّر أي شيء ، فالعنوان الفيزيائي لـ « نهاية » يسبق العنوان الفيزيائي لـ « بداية » .



شكل 4.3 : الداريء الدائري

3.2.3 . الداريء المزدوج (buffer)

هذه الأوالية هي الأسهل . يُضاف إلى القنال مجموعة من دارئين . خلال مدة عمل وحدة الإدخال ـ الإخراج على الداريء الأول (بواسطة BIOS) فإن المهمة المُستعمِلة تعمل على الامر (بواسطة LIOCS) .وعندما تفرغ إحدى ذاكرات الداريء من المعلومات فان الأخرى تقوم بشحن المعلومات ، ويتم التبادل . يتعلّق ذلك بحالة pool من ذاكرات الداريء ولكن مختزلة إلى إثنين منها فقط . إذا كانت المهمة المُستعملة سريعة جداً فسيتم وضعها عادة في إنتظار استعداد الداريء .

4 . طرق البلوغ

طريقة البلوغ تُعرِّف التنظيم المنطقي للمعطيات وتعزل المهام المُستعمِلة عن مواصفات العتاد . وبشكل عام ، من الممكن دائماً جعل نظام التشغيل مقتصراً على إجراء عمليات بلوغ بسيطة لوحدات الإدخال ـ الإخراج ؛ لن نتكلم أبداً عن عمليات البلوغ هذه ولكن سنفصّل لاحقاً تلك التي تحتوي على طبقة منطقية نموذجية للمستعملين . في هذا المفهوم ، يوجد ثلاثة أنواع بلوغ كلاسيكية .

- البلوغ المتسلسل Sequential access

- البلوغ المتسلسل المؤشر indexed sequential access

- البلوغ العشوائي Random access

1.4 . البلوغ المتسلسل

هذه التقنية تقوم ، بعد فتح السجل ، على التركيز على التسجيلة الأولى المنطقية الموجودة في هذا السجل . تُركَّز قيمة العدّاد الذي يُخزِّن رقم التسجيلة على « الصفر » ، بعد ذلك نزيد إلى قيمته واحداً في كل مرَّة نمرّ فيها على التسجيلة المنطقية التالية . طول التسجيلات المنطقية يمكن أن يكون متحولاً (سطر مثلاً) ، يكفي إذن لبلوغ نهايتها أن نضع سمة خاصة من نوع ASCII ، أو متتالية من السمات («عودة المنزلقة CR » ، « نهاية السطر LF » مثلاً) . هذه التقنية هي بسيطة وفعّالة ، وتُناسب السجلات المنطقية أكثر من بقية وحدات الإدخال - الإخراج الأخرى .

2.4 . البلوغ المتتالي المؤشر (Sequential indexed access)

عندما نرغب باستخراج معطيات معينة من سجلٍ منطقي مخزَّن على الذاكرة الخارجية ، فإن البلوغ المتتالي هو ثقيل لأنه يحتاج إلى عبور كامل السجل . من الممكن إذاً تخصيص كل تسجيلة منطقية بمفتاح . تجمع هذه المفاتيح في مؤشر يقوم بإجراء التناسب بين المفاتيح والعناوين الفيزيائية للتسجيلات . تُرتَّب التسجيلات حسب الترتيب الأبجدي والعناوين الفيزيائية المُناسبة تكون بترتيب تصاعدي . البلوغ عند القراءة لتسجيلة واحدة هو سريع جداً ، وعلى العكس فإن إدخال تسجيلة جديدة في وسط السجل هو غير ممكن إلا بإعادة كتابة المجموعة في هذا الأخير .

3.4 . البلوغ العشوائي Random access

توضع التسجيلات المنطقية بترتيب معين ، على العكس ، فإن طول هذه التسجيلات هو ثابت ويجب أن يتناسب مع طول التسجيلات الفيزيائية ، مثلاً : طول قطاع دائري (sector) من الأسطوانة .

العنوان المنطقي ، أو المفتاح ، المحدَّد بواسطة مهمة مُستعملة يسمح ببلوغ مباشر لموقع التسجيلة في السجل لأنه من الممكن أن نحصل منه مباشرة على رقم التسجيلة الفيزيائية . لا يوجد أي مفهوم للترتيب في السجل ، مما يسمح بعمليات قراءة وتعديل سريعة للتسجيلات المنطقية . على العكس يوجد بعض الهدر في

مساحة الذاكرة ناتج عن كون التسجيلات المنطقية لا يمكن أن تكون في أي حالة موزعة على عدة تسجيلات فيزيائية . هذه الطريقة هي مفيدة بشكل خاص لتسجيل مجاميع المعطيات (data base) .

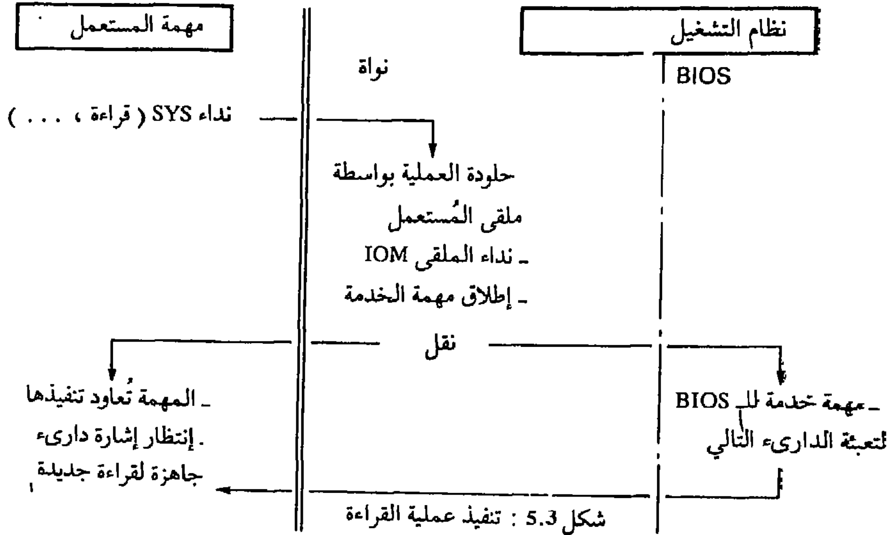
5 . أوالية تنفيذ طلبات الإدخال ـ الإخراج

1.5 . الحالة العامة

يقوم طلب الإدخال ـ الإخراج الجاري بواسطة مهمة مُستعمل معين على نداء نظام التشغيل (System service request: SSR) مستعملاً كقياس خدمة إدخال ـ إخراج معينة (فتح سجل للقراءة المتتالية مثلاً) . يُعالج هذا النداء في مستوى ملقى المستعمل (user interface) (أنظر الفصل الخامس) الذي يبدأ بتخزين نصّ المهمة المُنادية (على الأقل في نظام متعدِّد المهام) ، بعد ذلك يقوم بحلودة (فك رموز decode) متغيرات الطلب . لا تقوم أنظمة التشغيل متعدِّدة المهام بإيقاف المهمة المُستعملة ولكنها تقوم بإنشاء مهمة خدمة دورها إجراء العمل المطلوب الذي يُنفَّذ بشكل شبه ـ متواز مع المهمة المُستعمِلة . هذه الأخيرة يتم إعلامها بأن الخدمة قد إنتهت باستلام الحادثة . في حالة أنظمة التشغيل البسيطة ، يتم ذلك بواسطة المهمة المُنادية التي تقوم بتنفيذ إجراءات خدمة نظام التشغيل قبل إكمال تنفيذه .

جميع المراحل التحضيرية لتنفيذ عملية الإدخال ـ الإخراج تتم بواسطة مُنظّم الإدخال ـ الإخراج (IOM) . هذا الأخير يتحقَّق في كل عملية من تماسك وتنفيذ الخدمة المطلوبة . مثلاً ، عند طلب إجراء عملية قراءة لتسجيلة منطقية من خلال قنال معين ، فهو يتحقَّق من أن هذا القنال قد فُتِح جيداً للقراءة وبأنه قد تم فعلاً تخصيصه إلى المهمة التي ستقوم بتنفيذ الطلب . المعلومات الضرورية لهذا التحقُّق تُوجد في واصف القنال . بعد ذلك يتم تفصيل (بشكل مهام خدمة أو تنفيذ إجراءات) عمليات LIOCS وBIOS الضرورية . هذه الأخيرة تمرّ بواسطة مُنظِّم ذاكرات الداوىء في كل مرَّة تحتاج فيها إلى دارىء جديد ، أو إجراء نقل لدارىء فيما بينها (الداوىء المكتوب بواسطة BIOS يُنقل إلى LIOCS لكي تجري قراءته) ، أو إطلاق حرية دارىء . وفي النهاية عندما ينتهي كل شيء ، فإن مُنظِّم الإدخال ـ الإخراج يقوم بإرسال حادثة (إشارة) نهاية الخدمة إلى المهمة التي كانت قد طلبت الخدمة . هذه الأوالية هي مُوضَّحة مثلاً ، في حالة الشكل 2.3 ، ولقد افترضنا أن مهمة LIOCS كانت قد جرت بواسطة نداء بسيط لإجراء يستطيع وقف مهمة المُستعمل إذا لم يكن

هناك أي داريء جاهز في علبة الرسائل BPL ، بينما مهمة BIOS تتمّ بواسطة مهمة خدمة . مدير الداريء يُوضع في العمل مباشرة بواسطة منظّمات علبتي الرسائل BPL وBLP .



شكل 5.3 : تنفيذ عملية القراءة

2.5 . «Spooling»

هو عبارة عن تقنية إدارة وحدات الإخـراج غير القـابلة للتجزئـة ، كالـطابعة ، حسب مفهوم متعدِّد المستعملين . يجب أن يستطيع كل مُستعمل أن يطلب ، وفي أي لحظة ، طباعة سجل معين ، وذلـك دون أن تخلط الطابعـة بين الأسطر من مختلف السجلات ودون أن تُعطِّـل عمل أحد المستعملين خلال المدة التي يقوم فيها الآخـر بعملية الطباعة .

في هذا المفهوم ، فإن منظِّـم الإدخال ـ الإخراج (IOM) هو عبارة عن مهمـة تقوم ، بدلاً من إجراء عملية الإرسال المطلوبة بشكل مباشر ، بتحـويل المسـار إلى سجل فيزيائي في الذاكرة الخارجية . لكل مهمة تقوم بإجراء طلب لـلإرسال يجـري تخصيص سجـل مختلف لها . وفي كـل مرَّة ينتهي المُستعمـل من عمليـة الإرسـال الخاصة به ، يجري إغلاق السجل ووضعه في لائحة الإنتظار . وبشكل متواز ، في كل مرَّة تكون فيها وحدة الإخراج الفيزيائية (الطابعة) جاهزة ، فإن مُنظِّـم الإدخال ـ الإخراج المُتخصِّص (spooler) ينقل السجلات الموضوعة في الانتظار إلى الطابعة .

السيئة الوحيـدة لهذه الـطريقة هي في أنهـا تؤدي إلى استعمال مُكثّـف لقنـال الاسطوانة .

الفصل الرابع

تنظيم السجلات

1 . المفاهيم الأساسية

يتألف السجل من مجموعة من المعلومات مرتبطة بعلاقات معينة فيما بينها . وينقسم إلى مجموعات ثانوية تدعى تسجيلات . ويسمح بتخزين وبعد ذلك بإيجاد معلومات ليست بحاجة إلى الركون دائماً في الذاكرة الحيَّة (ذاكرة مركزية) بشكل دائم . نقوم إذاً بالإستعانة بالذاكرة الخارجية (أسطوانات ليِّنة ، اسطوانات صلبة ، أشرطة مغناطيسية) وعند ذلك نواجه مشاكل التوزيع : تقسيم الذاكرة الخارجية إلى قوائم وسجلات ، تقسيم السجلات بين المُستعملين . من هنا ضرورة حماية المعلومات بإيجاد حقوق لبلوغها .

يهدف تنظيم السجلات إلى تقديم بعض الخدمات للمستعملين . يتمتع هؤلاء بإمكانية بلوغ إلى المواصفات المنطقية النموذجية ، وليس أمامهم أية مشكلة أو إهتمام بمختلف النظم الفيزيائية المتخصصة حسب نوع الذاكرة الخارجية المستعملة . أما الخدمات التي يقدِّمها مُنظِّم السجلات فتسمح بشكل أساسي بحماية البلوغ إلى المعطيات (حماية ضد الأخطاء العفوية للعتاد بواسطة نسخ يومي وتخزين (back-up) وإدارة عمليات البلوغ المتعدِّد (قسمة السجلات)).

للقيام بهذه المهام ، هناك بعض المتطلبات التي من الواجب أن يؤديها منظِّم السجلات .

- يجب أن يستطيع كل مستعمل أن يقوم بإنشاء ، تعديل ، أو إلغاء للسجلات .
- يجب أن يستطيع كل مستعمل أن يعرف حقوق بقية المستعملين ببلوغ السجلات .

- يجب أن يستطيع كل مُستعمل ، حسب حقوق البلوغ الخاصة به ، أن يقرأ ، يُعدِّل ، يُعيد تركيب ونسخ سجلات غيره من المستعملين .
- جميع خدمات التخزين ومعاودة العمل يجب أن تكون بتصرف المستعملين في حالة وقوع حادثة معينة .
- بلوغ السجلات يجب أن يتم بواسطة إسم رمزي .

1.1 . القائمة أو الإضمامة (DIRECTORY)

لكل مستعمل لائحة بأسماء سجلات موزعة في قائمة أو في عدة قوائم أو إضمامة . ويعطي نظام تعدُّد المستعملين بشكل عام وأتوماتيكي إضمامة خاصة أو قائمة لكل مُستعمل تحمل إسمه ؛ بعد ذلك يصبح كل مستعمل حرّاً بإنشاء قوائمه الخاصة لترتيب وتركيب السجلات بفصل ، مثلاً ، السجلات القابلة للتنفيذ عن سجلات النصوص . هكذا ، فالقوائم هي عبارة عن سجلات خاصة مخزَّنة في الذاكرة الخارجية ومزوَّدة بحقوق بلوغ كبقية السجلات . تنظم هذه القوائم على شكل شجرة (tree) بجذع يمثل القائمة الرئيسية التي من خلالها نستطيع أن نصل إلى جميع القوائم الأخرى . كل قائمة هي عبارة عن فرع من هذه التركيبة الشجرية والسجلات عبارة عن الأوراق .

تتمتع القائمة (الإضمامة) الرئيسية بموقع ثابت معروف من مُنظِّم السجلات (بشكل عام المسار الأول من الاسطوانة) وينشأ عند وضع وحدة التخزين في الخدمة (تنسيق الاسطوانة مثلاً) . لبلوغ أحد السجلات يجب أولاً بلوغ القائمة التي تحتوي على هذا السجل . وعندما يبدأ أحد المستعملين بدورة عمله (session) على الحاسب (login) ، فإن نظام التشغيل يقوم بتركيزه على القائمة التي تحمل إسمه ، مما يجعل القائمة الرئيسية بتصرُّف المستعمل بقسمها الأكبر .

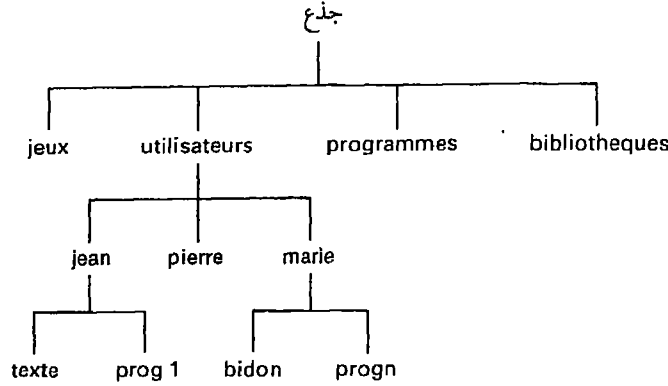
الشكل 1.4 يعطي مثلاً على التركيبة الشجرية للقوائم .

هذه التركيبة تحتوي على أربع قوائم . إن قائمة المستعملين تتضمّن ثلاث قوائم للمستعملين : Jean ، pierre ، maric . القائمة ماري (marie) تحتوي على سجلّين : bidon وprogn . بلوغ السجل bidon يجب أن يتم بشكل طبيعي حسب المسار التالي :

racine / utilisateurs / marie / bidon

عملياً ، فإن الأسماء الكاملة ليست ضرورية إلا عندما نرغب ببلوغ أحد

السجلات غير الموجودة في القائمة الجارية ، في الحالة المعاكسة يكفي إسم السجل فقط .



شكل **1.4** : شجرية السجلات

1.2 . واصف السجل (file descriptor)

يُضاف إلى كل سجل تركيبة معطيات أساسية تحتوي على عدد من المعلومات التي تسهّل تنظيمه . عدد هذه المعلومات يتعلّق بدرجة تعقيد نظام التشغيل . على سبيل المثال ، هذه هي لائحة بالمعلومات التي من الممكن أن تخزّن في واصف السجل :

- إسم السجل ،
- إطالة الاسم (extension) ،
- خواص ،
- إسم مُنْشىء السجل ،
- حقوق البلوغ ،
- تاريخ أخر تعديل في السجل ،
- تاريخ آخر تعديل للسجل عند القراءة ،
- العنوان الفيزيائي لأول تسجيلة على الاسطوانة ،
- حجم السجل ،
- عنوان (أو عناوين) القوائم التي يوجد عليها السجل ،
- الخ .

سنقوم الآن بتفصيل بعض هذه الحقول .

3.1 . مميزات السجل

ثلاثة عناصر أساسية من الواصف تستعمل لتحديد مواصفات السجلات وهي : الاسم ، الإطالة والخواص .

أ ـ الإسم :

ويُمثَّل بواسطة سلسلة من السمات (بعدد أقصى 8 في النظام CP/M وMS/DOS و14 سمة في النظام UNIX) يمكن أن تحتوي على أحرف ، أرقام وبعض السمات الخاصة ، يُستعمل الاسم كمعرِّف أساسي للسجل .

ب ـ الإطالة (extension)

وتسمح بالإشارة إلى طبيعة المعلومات الموجودة في السجل (نص ، برنامج مصدري ، كود قابل للتنفيذ ، الخ) . بعض الإطالات هي نموذجية ومعروفة من نظام التشغيل ، البعض الآخر حرّ ومعروف فقط من قِبل المستعمل . في النظام UNIX لا يوجد إطالات نموذجية بينما في النظام MS/DOS يوجد البعض منها : مثلاً :

ـ سجل قابل للتنفيذ	PROG1. COM
ـ سجل نصّي	TEXTE.TXT
ـ سجل من النظام	ANSI.SYS

ج ـ الخواص

وتدل على نوع السجل ، والطرق المستعملة للحماية ، الخ . يوجد لغو بين مفهوم الإطالة عندما تستعمل من قِبل النظام والخواص ، في بعض أنظمة التشغيل (UNIX مثلاً) ، فقط الخواص هي مستعملة مباشرة بواسطة نظام التشغيل ، والإطالات ليست معروفة إلا من البرامج المُساعدة من نوع مُصرِّفات . تحت إشراف UNIX تدل السمة الابجعددية على أن السجل هو :

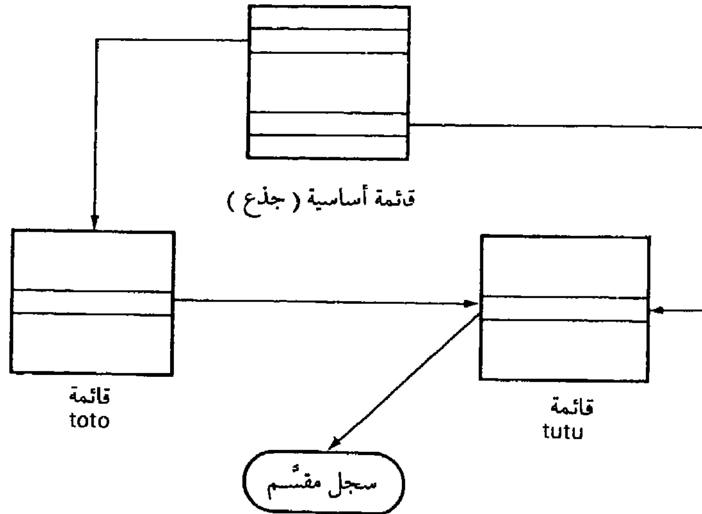
ـ سجل بسيط « ـ » ،
ـ قائمة « d » .
ـ سجل خاص يُناسب وحدة الإدخال ـ الإخراج «b» أو «c» .

4.1 . القسمة والحماية

يُخزِّن واصف السجل عدداً من المعلومات التي تسمح بقسمة السجل بين عدة

مُستعملين مع تأمين بعض الحماية لها . يمكن لمنشىء السجل أن يسمح لعدد من المستعملين باستعماله . تؤدي الحماية إلى مراقبة المحافظة على حقوق البلوغ . في النظام UNIX مثلاً ، يُجمع المُستعملين في مجموعات وتُحدَّد حقوق البلوغ لكل مستعمل حسب مجموعته ولكل نمط استعمال (قراءة : «r» ، كتابة : «w» أو تنفيذ : «x») .

من جهة أخرى ، وكي يتم تفادي عبور شجرة السجلات في كل مرَّة يتم فيها قسمة السجل ، من الممكن أن نجعل سجلاً معيناً موجوداً في عدة قوائم . بعد أن يتم إنشاء السجل ، يكفي إنشاء وصلة بين السجل وقائمة معينة (هذا غير ممكن إلا بشرط أن يكون الشخص الذي يقوم بإنشاء هذه الوصلات يتمتع بحق تعديل القائمة وقراءة السجل) . يظهر السجل إذاً في قائمتين بشكل متشابه بدون نسخ كما هو مُشار إليه في الشكل 2.4 .



شكل 2.4 : قسمة السجل

تسمح أوامر نظام التشغيل ببلوغ السمات الشائعة للسجلات وتعديلها . دائماً باعتماد المثل UNIX ، الأمر ll سيعطي جميع المواصفات الأساسية للسجلات من القائمة الجارية . مثلاً :

```
- r w - r w - r - -   1  toto  304 May 28 09:21 desal.p
- r w x r - x - - -   2  toto  560 Jun 24 10:08 desal
d r w x r w x r - x   2  toto  368 Sep  3 17:40 reper1
```

الحرف الأول يناسب خاصية نوع السجل («d» : إذا كان ذلك كقائمة مثل reper 1). بعد ذلك تأتي حقوق البلوغ لمُنشيء السجل (—r w أي حقه بالقراءة وبالتعديل ، ولكن لا تنفيذ لـ desal.p) ، بعد ذلك حقوق البلوغ للمستعملين المُنتمين إلى نفس مجموعة مُنشيء (creator) السجل (—rw لـ desal.p) وفي النهاية حقوق بلوغ بقية المستعملين (— —r تشير إلى أن بإمكان بقية المستعملين القراءة ولكن لا تعديل desal.p) . الرقم حسب تعريف حقوق البلوغ يعطي عدد الوصلات الموجودة بين السجل والقائمة . السجل desal.p لا يظهر إلا في القائمة الجارية بينما desal يظهر في قائمة أخرى . بعد ذلك يأتي إسم مُنشيء السجل (toto) ثم الحجم مُقاساً بعدد التسجيلات الفيزيائية المشغولة ، بعد ذلك يأتي تاريخ اخر تعديل (28 أيار الساعة 09 , 21 لـ desal.p) ، وبعد ذلك يأتي إسم السجل مع الاطالة (سيتم التعرف إلى desal.p كبرنامج مصدري بواسطة مصرف باسكال).

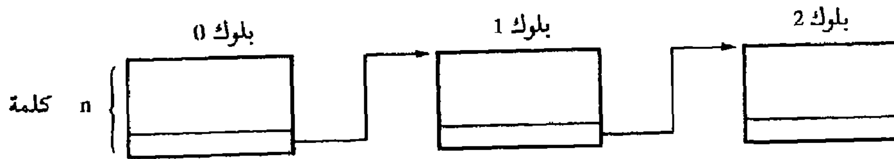
2 . تنظيم السجلات

يرتبط التنظيم الفيزيائي للسجلات بالمواصفات الفيزيائية لوحدات التخزين . أغلب وحدات التخزين تستعمل أشرطة مغناطيسية لا تسمح مثلاً إلا بالسجلات ذات البلوغ المتسلسل ؛ على العكس فإن استعمال الاسطوانات هو أكثر سهولة بسبب إمكانية الإنشاء الفيزيائية للاسطوانة في مسار وقطاع دائري . تناسب التسجيلات الفيزيائية على الاسطوانة بشكل عام القطاعات الدائرية . نلتقي إذاً بنوعين من التنظيم الفيزيائي للسجلات :

- متسلسل
- متلاصق
- مُقطّع

1.2 . التنظيم المتسلسل

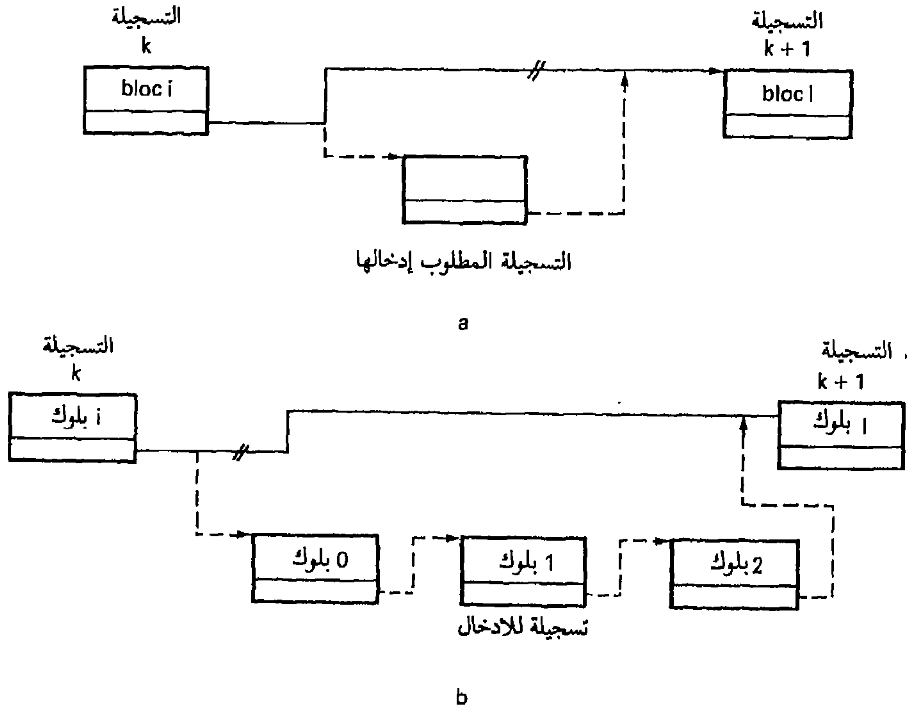
يتألف السجل المتسلسل من تسجيلات فيزيائية ، تدعى بلوكات ، متصلة فيما بينها بواسطة مؤشرات . يمكن للتركيبة المتسلسلة أن تكون بسيطة ، كما في الشكل



شكل 3.4 : التنظيم المتسلسل لأحد السجلات

3.4 ، أو مزدوجة (ربط مباشر ومعكوس) لتخفيض مدة تنظيم الحلقات (محو أو إدخال بلوكات جديدة) . في النظام CP/M ، عدد الكلمات المؤلفة من ثماني بتات في البلوك (الذي يتطابق مع القطاع الدائري (sector) هو 128 .

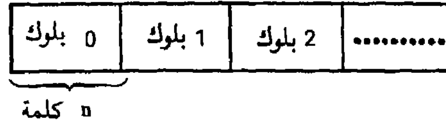
إدخال التسجيلة المنطقية الجديدة في السجل (البلوغ العشوائي (random)) هو عبارة عن عملية معقدة قليلاً حسب التناسب الموجود بين التسجيلات الفيزيائية (البلوك) والتسجيلات المنطقية . إذا كان هناك تطابق ، فإن الإدخال (insertion) هو سهل (شكل a.4.4) ويبقى كذلك إذا كانت إحدى التسجيلات المنطقية تحتوي على عدة تسجيلات فيزيائية (شكل b.4.4) . على العكس فهذا يصبح صعباً إذا كانت التسجيلة الفيزيائية تحتوي على عدة تسجيلات منطقية (حالة قليلة الاستعمال) .



شكل 4.4 : إدخال تسجيلة منطقية إلى السجل المتسلسل

2.2 التنظيم المتلاصق

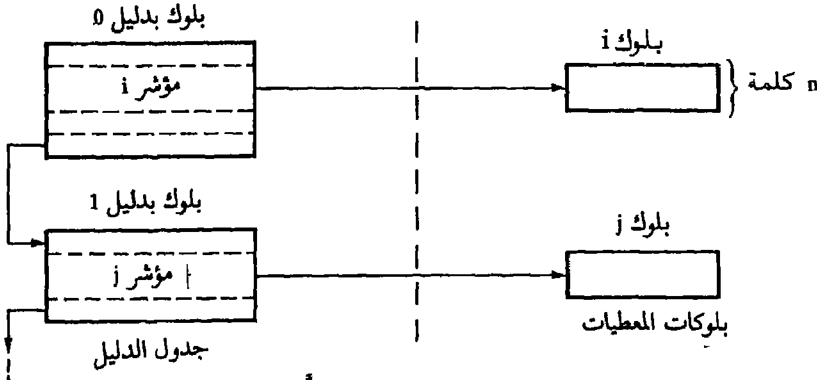
السجل هو متلاصق عندما يشكل قطاعات دائرية متتالية ومتجاورة على إحدى الاسطوانات ، أي إن التسجيلات الفيزيائية متلاصقة (شكل 5.4) . بلوغ المعطيات هو سريع جداً وأكثر سرعة من الحالة السابقة لأنه لا يوجد حلقة للعبور ، ولكن حجم السجل يجب أن يُثبَّت عند لحظة الانشاء . عمليات تسجيل تسجيلات جديدة ليست ممكنة إلا إذا كانت هناك مساحة تكفي قد جرى توقعها عند إنشاء السجل ، وإلا فسيحدث فيض . على كل حال فهي صعبة عادة لأنها تؤدي إلى تحريك جميع البلوكات حسب البلوك الداخل والجديد .



شكل 5.4 : سجل متلاصق

3.2 . التنظيم المُتقطِّع

يتألف السجل المتقطع من تسجيلات فيزيائية (بلوكات) بعنوان معروف بواسطة دليل (index) . نتكلم إذاً عن السجلات ذات الدليل . الدليل (أو جدول الدليل) هو عبارة عن مجموعة من مؤشرات البلوكات المنظَّمة بدورها بشكل متسلسل أو متلاصق في بلوكات الدليل (شكل 6.4) . بالنتيجة ، فإن البحث وإدخال بلوكات جديدة هو ممكن بشرط ، في حالة جدول الدليل المتلاصق ، أن نكون قد توقعنا مكاناً كافياً لإضافة الدلائل الجديدة فيه .



شكل 4.6 : السجل المقطّع

3 . مُنظِّم السجلات

تنظيم السجلات في أنظمة التشغيل « متعدّدة ـ المهام » يمكن أن يتم حسب إجراءين ممكنين عندما تقوم المهمة بطلب لإجراء عملية معينة :

ـ إما أن يضع مُنظِّم السجل المهمة بالانتظار حتى الإنتهاء من العملية .

ـ إما أن تتابع المهمة التي قامت بالطلب تنفيذه بشكل شبه متواز مع المُنظِّم ، وهذا الأخير يقوم بإرسال إشارة عند إنتهاء الخدمة .

الحلّ الثاني هو الأسهل ، ولكنه يفرض أن يتم إنشاء المهمة بواسطة نظام التشغيل لارجاع الخدمة له . الكود القابل للتنفيذ المُناسِب لمهمة الخدمة هذه يمكن إما أن يكون حاضراً في الذاكرة الحيّة (راكناً résident) وإما أن يكون مشحوناً من خلال الذاكرة الخارجية .

بشكل عام ، فإن طريقة عمل مُنظِّم السجل هي شبيهة بطريقة عمل مُنظِّم عمليات الإدخال ـ الإخراج المشروحة في الفصل الثالث . في أنظمة التشغيل البسيطة (غير متعدِّدة المهام) تتم الخدمة بواسطة تنفيذ الإجراءات «systèmes» التي يطلبها برنامج المستعمل . من الممكن أن نلتقي الحالات الثلاث :

ـ إما أن تكون المهام بأكملها راكنة في الذاكرة الحيّة ، أي أن تكون مشحونة مع نواة نظام التشغيل عند وضع النظام في العمل ،

ـ إما أن تكون موجودة مع نظام التشغيل غير المشحون في الذاكرة الحيّة إلا عندما يكون ذلك ضرورياً (over lay) ،

ـ إما أن تكون خارجية أي مع كل مهمة يناسب سجل قابل للتنفيذ شبيه بالسجلات القابلة للتنفيذ التي يقوم المستعملون بإنشائها ، يتعلّق ذلك ببرنامج مُساعِد من خارج المُنظِّم لتنظيم السجلات .

في مستوى الإستعمال ، الحالتان الأولى والثانية تتطابقان لأن تنظيم عملية التوافق (overlays) المحتملة هي بتصرُّف المستعمل ، على العكس فإن العمليات الخارجية هي مختلفة ، وعلى الأخص في حالة مُنظِّم الذاكرة الخارجية المركّبة من نواقل متحركة (اسطوانات مثلاً) . هكذا ، فبإشراف النظام MS-DOS نسخ السجل

في مكان آخريتم بواسطة العملية «COPY» الراكنـة والمبلوغة دائمـاً . على العكس فبإشراف النظام CP/M نفس المهمة ، تُدعى «PIP» وهي خارجية وليس من الممكن أن يتم إجـراء عمليات نسـخ إذا لم يكن بتصرُّفنـا سجـل يُـدعى «PIP.COM» على الاسطوانة التي في طور الاستعمال .

الفصل الخامس

ملقى المُستعمل
(User Interface)

ملقى المستعمل هو القسم « الخارجي » من نظام التشغيل . وهو يسمح للمستعمل بأن يتم التعرُّف إليه من قِبل النظام (Login) ، وبعد ذلك بتحديد حاجاته بواسطة الأوامر . الحوار بين الإنسان والآلة يتم إذاً بواسطة مجموعة أوامر على درجة متفاوتة من التعقيد ، يمكن أن يتم تجميعها ويمكن أن تخضع لقواعد نحوية . وهذا ما يؤدي إلى إدخال مفهوم لغة التحكُّم ، إما على شكل أوامر مُستقلة يجري إدخالها مباشرة بواسطة لوحة الملامس أو على شكل سجل من الأوامر ، يجري التعرُّف على كل أمر منه بواسطة مفسِّر خاص للأوامر (interpretor) .

هكذا فالاستعمال الفعَّال للنظام المعلوماتي ، تسبقه مرحلتان كلاهما بإشراف ملقى المُستعمل . إنّها التشكيل (configuration) وإعداد النظام (initialisation) . عملية التشكيل هي عبارة عن عملية لا تتم إلا عندما يتغيَّر عتاد النظام ؛ الإعداد ، نفسه ، يتم بعد كل عملية توقف للحاسب . في الأنظمة متعدِّدة ـ المستعملين ، عمليتا التعرُّف وإعداد المُستعمل (Login) تتمان في كلّ مرّة يُسمح فيها للمستعمل بالإتصال بالنظام . سنقوم باختبار المفهومين على التوالي .

1 . لغة التحكُّم

ترتكز هذه اللغة على مجموعة من الأوامر الموضوعة بتصرُّف المستعمل . هذه الأوامر تناسب إما عمليات ونظام التشغيل الراكنة في الذاكرة الحيّة ، وإما برامج خاصة قابلة للتنفيذ ومسجّلة في الذاكرة الخارجية ولكن مصنوعة من قِبل المستعمل نفسه.

حتى في نظام التشغيل الأسهل (نوع CP/M) ، هذه الأوامر يمكن أن تُجمّع بشكل سجلات قابلة للتنفيذ . تُفسَّر هذه السجلات سطراً بعد آخر بواسطة ملقى المستعمل . وحسب نظام التشغيل وتعقيده ، فإن هذه اللغة تحتوي على إمكانية إدخال متغيرات وسيطة عند نداء المتحولات وتركيبات التحكم المُركَّبة التي قد تؤدي إلى لغة حقيقية للبرمجة بنحو واضح .

في النظام MS-DOS تدعى هذه السجلات القابلة للتنفيذ سجلات «Batch» (extension. bat) . نجد فيها تركيبات تحكُّم من نوع «FOR» ، «GOTO» ، «IF» ، «PAUSE» ، الخ . لنفترض بأننا نرغب بتنفيذ البرنامج المدعو «monprog» عدة مرات مع التحكُّم في كل مرَّة بمعاودة التنفيذ ، هذا يمكن أن يتم بواسطة السجل التالي «montfich.bat» :

```
ECHO off
: debut
monprog
ECHO pour sortir appuyer sur ctrl-c
PAUSE
GOTO : debut
```

للخروج نضغط على الزر ctrl-c

بداية

بعد كل تنفيذ للبرنامج «monprog» فإن الرسالة التالية تظهر على الشاشة : (للخروج يجب الضغط على الزر ctrl-c)

```
pour sortir appuyer sur ctrl-c
Strike any key when ready
```

لو إفترضنا بأننا ضغطنا على زر آخر غير ctrl-c فإن البرنامج «monprog» سيتم تنفيذه مرة أخرى .

في النظام UNIX فإن لغة التحكم («Shell») هي شديدة الفعالية وتحتوي على إنشاءات مركبة . نفس المثل السابق سيعطينا .

```
reponse=o
WHILE TEST $reponse=o
DO
monprog
ECHO 'voulez-vous continuer o/n'
READ reponse
DONE
```

السمة «0» تُعطى للمتحولة «reponse» ، بعد ذلك يتم تنفيذ سلسلة التعليمات الموجودة بين DO و DONE طالما إن المتحولة «reponse» تعادل السمة «0» . العملية READ تقرأ سلسلة السمات وتعطيها للمتحولة «reponse» .

2 . التشكيلة (configuration)

رأينا أن عملية تشكيل نظام التشغيل على مكنة معينة لم تكن لتتم لولا إمكانية تعديل محيط العتاد . مثلاً ، تشكيلة النظام UNIX لمكنة تحتوي على أسطوانة قاسية ستقوم على تقسيم هذه الأخيرة إلى عدد من الأسطوانات الفرضية ، وتحديد عدد الأدوات الطرفية المرتبطة بهذه المكنة إضافة إلى مميزاتها وفي النهاية تعريف فئات المستعملين وفي كل طبقة أسماء المُستعملين المسموح لهم بالعمل . كل هذا يتم بالعمل بواسطة نواة « دنيا » لنظام التشغيل تسمح بالعمل بطريقة المُستعمل ـ الموحَّد ، مثلاً : خرطوشة متحركة تسمح بشحن النظام . كل من عمليات التشكيل هذه تتم بإدخال عدد من المعلومات في السجلات المحدَّدة .

عملية تقسيم الاسطوانة القاسية إلى حيزات مُعتبرة كأسطوانات فرضية ، تهدف إلى الفصل ، مثلاً :

- خزن على إسطوانة نظام التشغيل .
- خزن البرامج المُساعدة مع المصرِّفات .
- خزن صورة الذاكرة الحيّة عند إجراء «Swaps» أو عند إطلاق منظِّم الذاكرة الفرضية في العمل .
- خزن برامج المستعملين .

المعلومات المطلوب إدخالها تتعلَّق بحجم مختلف هذه الحيزات ، وموقعها على الأسطوانة القاسية ، وتخصيصها إلى جذوع الشجرة لقوائم مناهج الاسطوانة القاسية .

تعريف الأدوات الطرفية المتصلة بالمكنة يقوم على تعريف الأسماء المنطقية لوحدات الإدخال ـ الإخراج (الأسماء التي ستسمح بتعريف السجلات الخاصة المناسبة لهذه الوحدات) مع تحديد فيما إذا كان ذلك يتعلَّق بالأجهزة المحيطية ، كالطابعات مثلاً أو بالأدوات الطرفية العملية (القنصلة التي تسمح للمستعمل بالعمل) . بشكل عام فإن الأسماء المنطقية المُستعملة هي مُخصَّصة لوحدات الإدخال ـ الإخراج الفيزيائية المحدَّدة . مثلاً ، بإشراف النظام UNIX ، فإن الخط :

12 tty 13

من السجل etc/ttys/ يعني أنه بامكاننا تعليق قنصلة على الخط 3 ، وهذه القنصلة سينظر إليها المستعمل وكأنها سجل خاص بالاسم dev/ttyl 13/ .

بعد ذلك ننسب إلى كل إسم منطقي إسماً جديداً من نوع جهاز محيطي في السجل etc/ttytypes/ مثلاً

tty 13 : v5

الذي يعني اننا غلّقنا على الخط 3 قنصلة من نوع «v5» . أما وصف جميع مميزات القنصلة من نوع «v5» ، فسيتم إعطاؤه في السجل etc/termcap/ . بهذه الطريقة فإن منظّم وحدات الأجهزة المحيطية لنظام التشغيل سيعرف جميع أوامر العرض الموضوعة بتصرُّف المستعملين (محو الشاشة ، تحريك الإشارات الضوئية (cursor) إلى اليمين أو الى اليسار ، الخ . .) .

وفي النهاية فإن السجل etc/group سيحتوي على لائحة بالمُستعملين المُخوَّل لهم بالعمل بواسطة المجموعة ، والسجل etc/passwd/ سيُحدِّد بعض مواصفات المُستعملين .

مثلاً :

- كلمة العبور (password)
- رقم المعرِّف المضافة إلى النظام ،
- عقدة شجرية القوائم المخصَّصة للنظام (القائمة الأولية للعمل) ،
- إسم المهمة الأولية المطلوب تنفيذها عند التعليق (يتعلَّق ذلك بمفسِّر لأوامر نظام التشغيل) .

3 . الإعداد (initialization)

يتم الإعداد عند كل عملية وضع للحاسب في الخدمة وربطه بالتيار بعد توقفه عن العمل . ويقوم على التعرُّف على العتاد بفحص موقع الميكرو- مفسِّرات . هكذا نُحدِّد كمية الذاكرة الممكن التصرُّف بها ، ووجود أو عدم وجود لاسطوانة قاسية ، بطاقات إدخال - إخراج الخ . بعد ذلك ، فإن الحاسب يقوم بتنفيذ السجلات المحدَّدة عند التشكيل (أو يقوم بتنفيذ مهام نظام التشغيل التي تستعمل سجلات التشكيل كمعطيات) لتكملة التعرُّف على المحيط .

عنـدمـا يتم إجـراء الإعـداد الأولي للمكنــة ، يمكن للمستعملين أن يتصلوا بالنظام . في حالة النظام « موحد ـ المستعملين » فإن مُفسِّـر الأوامر يصبح فعالاً . في حالة نظام تعدُّد ـ المستعملين (UNIX مثلاً) فإن الاعـداد ينتهي بعد تنفيـذ المهمة «Login» على جميع القناصل المصرَّح عنها كفعّـالة . هدف هذه المهمة هو التعرُّف على المستعملين ، وطلب كلمة العبور وبعد ذلك إطلاق مُفسِّـر الأوامر في العمل . بـإمكـان كـل مُستعمـل ، بشكـل عـام ، أن يقــوم بـإنشــاء سجـل أو إسم مُحـــدَّد («autoexec.bat» في النـظام MS/DOS أو «profile» في النظام UNIX) مُبـ في لغة التحكُّـم ويتم إطلاقه أوتوماتيكياً بواسطة مفسِّـر الأوامـر عند كـل عملية إ . وهذا يسمح بتحديد محيط العمل بإطلاق ، مثلاً وبشكل أوتوماتيكي ، نظام مع ة النصوص في السكرتاريا .

4 . البرمجة

عنـدما ينتهي الإعـداد ، يقوم المستعمِلون الـذين يرغبـون بصناعـة بـرامجهم التطبيقية باختيار لغة للبرمجة تسمح بإخراج الكـود القابـل للتنفيذ على مكنـة فرضيـة محدَّدة جزئياً بواسطة العتاد ونظام التشغيل الموضوع في التنفيذ .

يجب أولاً الإشارة إلى أن أنظمة التشغيل ليست مكتوبة بالكامـل بلغة التـأويل (assembler) . يكتب فقط نواة صغيرة منها ، بعد ذلـك حول هـذه النواة يتم تـطوير المناهج بـاستعمال لغـة للبرمجـة بمستوى عـال (High level) . مثلاً ، جـرى كتابـة UNIX باللغة C . هذا لا يكون بدون أية فائدة بالنسبة للمستعملين ، فإستعمال أوامر نظام التشغيل في داخل برنـامج مكتـوب بلغة أخـرى « بمستوى عـال » يتطلب عـادة العبور من لغة إلى أخرى وهذا ليس سهلاً دائماً . لهذا السبب نعتمد عادة عدداً مُحدداً من عمليات نظام التشغيل المبلوغة مباشرة من قِبـل المستعمل ، وذلـك عندمـا نقوم بالبرمجة بلغة ذات مستـوى عالٍ كلغـة PASCAL أو FORTRAN . بشكـل عـام ، يتعلَّـق ذلك بفتح السجلات المتتاليـة لإجراء عمليـات الكتابـة والقراءة منهـا وفيها . إضافة لذلك فهذه العمليات ليست نموذجية في لغة معينة على نـظام تشغيل معين . عندما نرغب باستغلال إمكانيات نظام التشغيل في داخل أحد البرامج ، يجب أن نقوم بالعمل في نفس اللغة وهذا أفضل من اللغة التي جرى إستعمالها لإنشاء هذا الأخير . مثلاً يجب العمل في لغة C بإشراف UNIX أو في PL/M بإشـراف iRMX 86 . لهذا نتكلم عن اللغة المميَّـزة لنظام تشغيل معين .

من الممكن أيضاً أن نلتقي الحالة المعتمدة للعمل في مستويات متدنية (أي في اللغات القريبة من العتاد مثل لغة المؤول أو لغة الآلة) كلغة FORTH مثلاً ، كما أنه باستطاعتنا إستعمالها على أنظمة لم يُكتب نظام التشغيل فيها . هذه اللغات تعرض مهاماً وعمليات تسمح بالعمل في المستوى المطلوب من نظام التشغيل . لهذا فالمفسِّرات FORTH العاملة على المُعالجات Z80 بإشراف CP/M تأخذ بعين الاعتبار موضع نظام التشغيل مما يجعل السجلات المُنْشأة في هذه اللغة غير مبلوغة بإشراف CP/M . هكذا فالمُفسِّر FORTH يحتوي نفسه على طبقة من نظام التشغيل بسيطة خاصة به وتسمح له بالعمل بشكل فعال على مكنات فرضية متكيفة بشكل جديد . كما نلتقي بهذا النوع من الأواليات في حالة لغات مفسِّرة أخرى مثل LISP .

إختيار لغة البرمجة تتعلَّق بمحيط البرمجة الموضوع حول نظام التشغيل أي مجموعة البرامج المُساعدة الموضوعة بتصرُّفنا :

- المنقحات (editors)
- المصرِّفات (compilers)
- المكتبات الخاصة (libraries)
- مُساعد للوضع في العمل (debuggers)
- منقِّح الأربطة (linkers)
- الخ .

هذه البرامج المُساعدة ترتبط مباشرة بنظام التشغيل نفسه إذا لم تكن مستقلة . مثلاً ، من الواضح أن القسم « مُصرِّفات » يُعالج عمليات قراءة وكتابة وهي مرتبطة بنظام التشغيل أكثر منه باللغة المعتمدة . في مستوى عمليات الإدخال ـ الإخراج البيانية للرسوم ، من الشائع حالياً أن مفهوم الأداة الطرفية للرسم البياني ليست شائعة الاستعمال كثيراً بسبب ثقلها وكلفتها ؛ لذا فإن البرامج المُساعدة الموضوعة لهذا الغرض لا ترتبط بنظام التشغيل فقط ولكن بالعتاد المُستعمل .

الفصل السادس

النظام CP/M

1 . مدخل

CP/M يعني Control program for microprocessor (برنامج تحكُّم بالميكروبروسسور).هو من إنتاج شركة Digital research ، ظهر سنة 1974 ، وعُرِض بشكل تجاري سنة 1975 . ولقد أصبح وبسرعة نموذجاً عن أنظمة التشغيل للحاسبات الصغيرة بكلمة 8 بتات (intel 8080- 8085, Zilog Z80) . ولقد جرى نشر صيغة جديدة تُدعى CP/M86 للحاسبات الصغيرة بطول 16 بتة للكلمة .

2 . تركيبة النظام CP/M

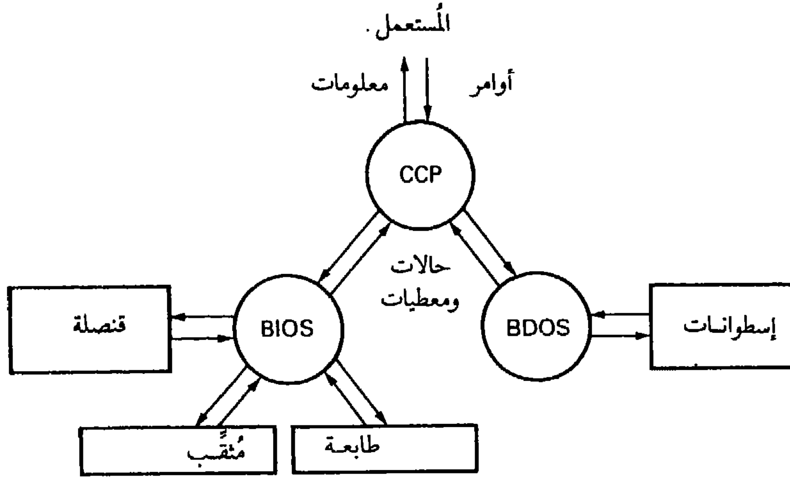
يتألف CP/M من ثلاث زُجل (module) :

- — CCP : Console Command Processor,
- — BIOS : Basic Input-Output System,
- — BDOS: Basic Disk Operating System.

CCP هو عبارة عن الملقى مع المُستعمل . يلعب دور مُفسِّر الأوامر ، وإضافة لذلك ، فمن مهامه تنفيذ هذه الأوامر ، الراكنة منها وغير الراكنة .

BIOSعبارة عن برنامج لتنظيم عمليات الادخال الاخراج الفيزيائية. ويتألف من مجموعة من الإجراءات التي تتصل مع الوحدات الفيزيائية المرتبطة بالنظام (قنصلة ، طابعة ، قارىء الاسطوانات) . يصبح البرنامج BIOS فعالاً بواسطة CCP وبمتغيرات تحدِّد له المهام المطلوب تنفيذها . فهو يؤدي مهام BIOS كما حددناها في الفصل الثالث .

BDOS هو عبارة عن زجلة من السجلات على الاسطوانات (أو على الاسطوانة القـاسية) . يؤمّـن للمُستعمـل إمكانيـة التعرّف على المميـزات الفيزيـائية لـوحدات التخزين بواسطة مجموعة من البرامج المُساعدة له والعمليات **الأساسية**، التي تتعلّـق تحديداً بالتعرُّف على موقع بلوكات المعطيات الموزعة على الاسطوانـة ، وصلاحيـة حقوق البلوغ وتكامـل المعطيـات . لا تجري معـالجة عمليـات الإدخال ـ الإخـراج الفيزيائية نحو القناصل والطابعة كسجلات خاصة والبرنامج BDOS يلعب دور منظّـم السجلات كما عـرضناه في القسم الأول ولكن يؤمّـن أيضـاً بعض عمليات BIOS . تفاعل الزجل الثلاث واتصالها بالوحدات الفيزيائية نراه على الشكل 1.6 .

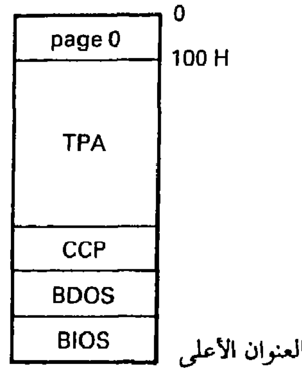


شكل 1.6 : تركيبة CP/M

3 . تشكيلية الذاكرة

تقسّـم الذاكرة الحيّـة للميكروحاسب بواسطة CP/M إلى خمسة حيزات كما هو ظاهر في الشكل 2.6 :

- حيّـز من 256 بايتة (000 إلى OFF بالنظام السادس عشري الذي نرمز إليه عـادة : OH وFFH ، «H» ترمز إلى الأرقام السادس عشرية) ، موضوعة في أسفل الذاكرة محجوزة للنظام ، وتدعى الصفحة 0 .



شكل 2.6 : تشكيل الذاكرة CP/M

- حيّز محجوز لبرامج المستعملين ويُدعى TPA (Transient program Area) الذي وحسب سعة ذاكرة الحاسب سيمتد من العنوان 100H الى العنوان 2900H لـذاكرة بحجم 16K بايتة ، أو إلى العنوان

$$2900H + n \times 4000H$$

حيث n هو عدد البلوكات من 16 K بايتة المضافة إلى الذاكرة ،

- CCP ،
- BDOS ،
- BIOS .

يُمثّل مضمون الصفحة 0 على الشكل 3.6 . نجد فيه متغيرات النظام . في البايتات الثلاث الأولى نجد تعليمة تفريع في BIOS إلى نقطة الدخول للإنطلاق (أنظر الفصل الخامس) . العنوان 3 المدعو بايتة IOBYTE يدل على التخصيص الجاري للأجهزة المحيطية (قارىء الأسطوانات ، قنصلة أو حاسبات) .

العنوان 4 يُحدِّد وحدة الأسطوانات العاملة ورقم المُستعمل . العناوين 5 ، 6 ، 7 تحتوي على تعليمة التفرُّع إلى البرنامج BDOS . عند كل عملية تضع القنصلة ، الطابعة أو الأسطوانة في العمل ، تجري دعوة النظام بواسطة العنوان رقم 5 . من العنوان 8 إلى العنوان 5CH (غير داخل) نجد الحيز المدعو « معاودة العمل » الذي يُطلب بواسطة التعليمات RSTO إلى RST7 من 8080 . من العنوان

5CH إلى العنوان 80H نجد واصف السجل نحو النقصان : (Default File Control Block DFCB) الذي هو عبارة عن بلوك من 32 بايتة يجري إختيارها بواسطة CCP بشكل أوتوماتيكي إضافة إلى حيز التوسيع من FCB (واصف السجل) لعمليات البلوغ المباشرة .

المساحة من الذاكرة من 80H الى FFH عبارة عن داريء (buffer) من 128 بايتة تحتوي على الأمر الجاري وتُشكِّل حيز الانتقال نحو النقصان للمعطيات نحو أو من خلال الاسطوانات .

المحتوى	العنوان
	FFH
داريء Buffer	
	80 H
DFCB	
	5CH
حيز المراجعة	
	8 H
التفرُّع إلى BDOS	
	5 H
وحدة الاسطوانات رقم المُستعمل	
	4H
IOBYTE	
	3H
تفرُّع إلى BIOS	
	0H

شكل 3.6 : الصفحة 0

كل دعوة للنظام من خلال برنامج المُستعمل تتم بالنقل بواسطة العنوان 0005H الموجود في الصفحة 0 . يصحب الدعوة متغيِّر يُحدِّد طبيعة الخدمة المطلوبة ، وهو متغيِّر منقول كرقم عملية (36 رقماً جاهزاً في الطبعة 2.2 من CP/M) ..

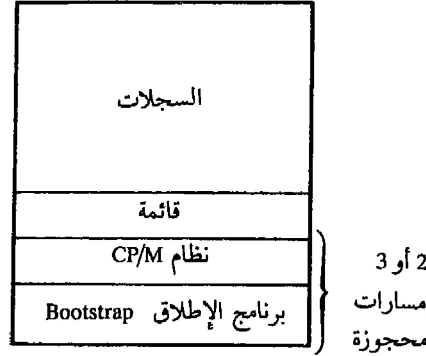
4 . السجلات

1.4 . التركيبة الفيزيائية

من الناحية الفيزيائية ، يتألف السجل من وحدات (حتى 16) ، كل وحدة تجمع تسجيلات فيزيائية (records) . تتألف كل تسجيلة من 128 بايتة (قطاع دائري في الاسطوانة) وكل وحدة يمكن أن تحتوي على 128 تسجيلة تناسب الحجم 16K بايتة للوحدة . يدير النظام CP/M الوحدات أوتوماتيكياً ، ويمكن للسجل أن يصل إلى 256 K بايتة .

يُضاف إلى كل سجل واصف (descriptor) أو FCB (File Control Block)

بتركيبة مُوضَّحة في الفقرة 3.4 . تُخزَّن مجموعة واصفات السجلات من الاسطوانة في منطقة خاصة من الاسطوانة متناسبة مع القائمة (directory) . الشكل 4.6 يُوجز توزيع المعلومات على الاسطوانة . يُنسخ واصف السجل في الذاكرة الحية (في الحيز TAP لأن الصفحة 0 لا تحتوي إلا على FCB نحو النقصان) في كل مرَّة يقوم فيها أحد الأوامر بطلب هذا السجل (مثلاً أمر بفتح السجل) .



شكل 4.6 : هيكلية الاسطوانة

2.4 أنواع السجلات

في النظام CP/M ، نشير إلى كل سجل بواسطة إسم يحتوي على 8 سمات على الأكثر متبوعة بنقطة وبإطالة مؤلفة من ثلاث سمات . تعني الإطالة نوع السجل . كما رأينا في القسم الأول ، فإن هذه الإطالة تنظَّم بواسطة النظام واما بواسطة المُستعمل .

الإطالات التي نلتقيها بواسطة النظام هي :

- COM سجل قابل للشحن والتنفيذ ،
- ASM سجل مصدري بلغة المؤول (assembler) ،
- PRN سجل لائحي للبرنامج المؤول (PRINT) ،
- HEX سجل برنامج بلغة الآلة قابل للشحن بواسطة CP/M ،
- BAK سجل خزن يُنشأ بواسطة المُنقِّح ،
- SUB سجل من نوع « لائحة الأوامر » قابل للتنفيذ بواسطة الأمر SUBMIT ،
- $$$ سجل مؤقت للعمل يُمحى بواسطة النظام .

هناك خاصية تحدِّد حق البلوغ وتحدِّد نوع السجل . قيم هذه الخاصية الممكنة

هي :

- (R/O) قراءة فقط (Read only) ،
- (R/W) قراءة كتابة (Read /Write) ،
- (SYS) سجل نظام ،
- (DIR قائمة (directory) .

3.4 . واصف السجل

واصف السجل FCB (file control block) يمكن أن يُمثِّل ويُعرِّف سجلاً بطول حتى 16K بايتة ، أي سجلاً لا يحتوي سوى على « وحدة » . كما رأينا فواصفات السجلات هذه هي مخزنة على الأسطوانة في القائمة ولكن يجري نسخها في الحيِّز TPA (الطبعة LIOCS مختزلة إلى التعبير الأسهل ، قسم من مُنظِّم ذاكرات الداريء يجب أن يتم بواسطة برامج المستعملين) من الذاكرة الحيَّة (RAM) عندما تكون في طور الاستعمال . وهذه هي النسخة التي يجري تعديلها خلال معالجة السجل . عند إغلاق السجل فإن الصيغة الجديدة من الواصف يجري نسخها على الأسطوانة في القائمة وذلك بمحو الصيغة القديمة .

يتألف واصف السجل من 32 بايتة. 16 بايتة الأولى تحتوي بالتحديد على إسم السجل مع الإطالة إضافة إلى حجمه. الـ 16 بايتة التالية تحتوي على عناوين القطاعات الدائرية المستعملة .

5 . التشغيل

العمل على البارد (cold start) (أي منذ البداية) يُناسب عملية إعداد النظام . عند إطلاق النظام في العمل على البارد (منذ البداية) يقوم البرنامج المدعو «boot strap» الموجود في الذاكرة ROM بشحن البرنامج (chargeur) « الشاحن » الموجود على الاسطوانة المتصلة بالنظام والعاملة في الذاكرة الحيّة . بعد ذلك تُشحن البرامج BDOS وBIOS وبرنامج التحكم بالوحدات BIOS التي تملأ في الصفحة 0 العناوين من 0 إلى 7 شكل (3.6) . من هذه اللحظة يكون النظام CP/M جاهزاً لاستقبال أوامر المؤثر (operator) ويعرض الرسالة A> . إرسال الأوامر يتم بواسطة BIOS (لأنه ينظِّم عمل القنصلة) في الداريء CCP . إذا كان الأمر راكناً ، فسيتم تنفيذه مباشرة وإلا سيتم إطلاق نداء إلى البرنامج BDOS لايجاد السجل القابل للتنفيذ المناسب والموجود على الاسطوانة وشحنه في TPA . ينسخ البرنامج CCP ، في

نفس اللحظة ، « واصف السجل » في الـذاكرة . وعنـدما ينتهي البـرنامـج CCP من عمله ، يُحرِّر المساحة من الذاكرة التي كان يشغلها ، ويأتي البـرنامـج TPA ليشغل هذه المساحة من بعده . الأمـر SUBMIT يسمح بتعليق لائحـة الأوامر المخـزَّنة في سجل بإطالة هي عبارة عن «SUB» .

الإطلاق على الساخن «Warm start» يتم بضـرب السمة «control C» على لوحة الملامس . وهذا الأمر يسمح بقطع وتدمير البرنامج الجاري ، ويُنصح به عندما نقوم بتغيير الاسطوانة على الوحدة . هذا الاطلاق يؤدي إلى إعادة إعداد الصفحـة 0 وإعادة شحن البرامج BDOS وCCP بدون تغيير صورة BIOS المشحونة في الذاكرة .

6 . أوامر النظام CP/M

1.6 . الأوامر الراكنة

وعددها ستة :

- TYPE nomfichier : يعرض لائحة مضمون أحد السجلات في الكود ASCII على القنصلة ،
- DIR : يعرض لائحة بأسماء السجلات الموجودة على الاسطوانة ،
- ERA nomfichier : يمحو سجلاً واحداً أو عدة سجلات ،

SAVE p nomfichier : يُخزِّن حيِّزاً من الذاكرة مؤلّفاً من p صفحـة (تناسب كـل صفحة 256 بايتة) في السجل nomfichier ،

- USER n : يسمح بتخصيص إسطوانة إلى عدة مستعملين يتميّـز كـل منهم برقم معيّـن $(0^2 \leqslant n^2 \leqslant 15)$. لا يسمح لكل مستعمل إلا ببلوغ سجلاته فقط .

2.6 الأوامر غير الراكنة

هـذه الأوامر تنـاسب سجلاً مـوجوداً في الكـود القابـل للتنفيذ المـوجـود على الأسطوانة المـوجودة على الـوحدة الفعَّـالـة وإلا ستكون هـذه الأوامر مجهـولة . من الممكن إضافة بعض البرامج المُساعدة للأوامر النموذجية أعلاه .

D: : يجعل الوحدة D فعالة ،

STAT: : أمر متعدِّد يسمح بـ :

- معرفة حالة الاسطوانة التي يُمكن أن تكون محميّـة عند الكتابة أو غيـر محمية وتغيير حالتها ،

- معرفة الموقع الموضوع بتصرفنا في الاسطوانة ،
- معرفة خاصيات السجلات وحجمها ،
- تعديل خاصيات السجلات .

PIP : (peripheral interchange program) أمر بنسخ ونقل السجلات من وحدة إدخال - إخراج إلى أخرى (قارىء الاسطوانات ، الطابعة ، القنصلة ، . . .) ، .

SYSGEN : أمر يسمح بإجراء نسخة للنظام CP/M على أولى مسارات الاسطوانة ،

MOVCPM : أمر يسمح بتشكيل صيغة CP/M لتكييفها مع تشكيلة الذاكرة المختلفة ،

ED : مُنقِّح الأسطر العاملة على سجلات بإشراف النظام CP/M ،

ASM nomfichier : برنامج مؤول لسجل مصدري مكتوب بلغة المؤول 8080 أو Z 80 لإنشاء كود مستهدف (سجل بإطالة «HEX») ،

LOAD nomfichier : يقلب سجل «HEX» إلى سجل «COM» قابل للتنفيذ مباشرة في الحيِّز TPA (تحويل العناوين المنطقية إلى عناوين فيزيائية متكيفة مع حجم الحيِّز TPA) ،

nomfichier : يطلق برنامجاً موجوداً في السجل «nomfichier» في التنفيذ ، إذا كان هذا الأخير من نوع «COM» ،

DDT nomfichier : برنامج «debug» للسجل nomfichier يسمح بتنفيذه بإشراف CCP ،

SUBMIT nomfichier : تنفيذ سجل لائحة بالأوامر «SUB» ،

DUMP nomfichier : يعرض على الشاشة بالتعبير السادس عشري مضمون السجل .

XSUB : أمر يسمح بإطالة إمكانيات الأمر SUBMIT بإنشاء أوامر قابلة للتنفيذ بإشراف SUBMIT .

الفصل السابع

النظام MS / DOS

1 . مدخل

جرى تعريف التـركيبة الأسـاسية للنـظام MS/DOS في سنة 1980 على شكـل النظام QDOS المُخصَّص للحاسبات الصغيرة 100 - S من إنتاج شركة Seattle com- puter products . وأصبح DOS -86 في نهايـة 1980 وبعد ذلـك تم شراؤه وتحسينـه بواسطة شركة Microsoft حيث أخذ الإسم MS-DOS . وقد أدى تكييف هذا النظام مـع حاسبـات IBM-PC الشخصية الى جعله نـظامـاً نمـوذجيـاً لتشغيـل الحـاسبـات الشخصية المرتكزة على الميكروحاسبات INTEL من العائلة 8086 و8088 .

كالنظام CP/M ، هذا النظام هو موجه إلى الأنظمة مُوحَّـدة ـ الاستعمال ، فهو إذاً غير متعدِّد المهـام . يستعمل عـدة أوامر تـأتي من النظام CP/M ولكنـه يضع في العمل مفاهيم وتصورات أكثر تطوراً في مستوى تنظيم السجلات المستوحاة من النظام UNIX مـع بقائهـا الأسهل . وقـد خضع النـظام MS-DOS ، كبقيـة المنـاهـج إلى تحسينات متتالية . هذه الإشارات التالية مأخوذة عن الصيغة 2 .

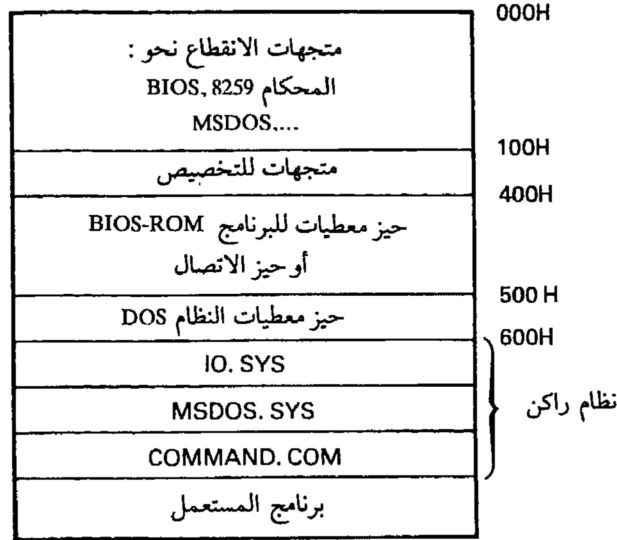
2 . مركّبات النظام MS-DOS

يرتكز النظام MS-DOS على نواة مزروعة في الذاكرة المركزية ، تدعى BIOS وترتبط بالعتاد . يلعب البرنامج BIOS ، إضافة إلى تنظيـم الإدخال ـ الإخراج ، دور إعداد الحاسب عند ربطه بالتيار ، والتحقّق من صحة عمل جميـع مركّبـاته (وحـدة مركزية ، ذاكرة ، أجهزة محيطية) ، وتنظيم الوقت (الساعة والتاريخ) ، وفي النهاية السماح بإجراء عمليات نسـخ للشاشـة . فهو يحتـوي إذاً على أشياء أخـرى أكثر من

البرنامج BIOS الذي عرفناه في الفصل الثالث .

مركبات النظام MS-DOS وعددها 4 :

- برنامج الإطلاق في العمل (Bootstrap) ،
- البرنامج IO.SYS ،
- البرنامج MSDOS.SYS ،
- البرنامج COMMAND.COM ،



شكل 1.6 : زرع النظام MS-DOS في الذاكرة

يقع برنامج الإطلاق على القطاع الدائري الأول من الاسطوانة الداخلة في الوحدة A عند ربط الجهاز بالطاقة . يُشحن النظام في الذاكرة بواسطة BIOS وذلك عندما ينتهي هذا الأخير من إجراء عمليات الاختبار لحسن سير عمل الجهاز . وبعد ذلك يقوم برنامج الاطلاق ببلوغ الاسطوانة من جديد لقراءة السجلات IO.SYS, MSDOS.SYS وCOMMAND.COM فيها . إذا كانت هذه السجلات غير حاضرة على الأسطوانة ، يطلب برنامج الاطلاق إدخال الاسطوانة «system» . يجب الإشارة إلى أن السجلات IO.SYS وMS DOS.SYS ليست ظاهرة عندما نعرض قائمة الاسطوانات ولا يمكن نسخها على اسطوانة أخرى إلا بعد تنسيق هذه الأخيرة .

يحتوي السجل IO.SYS على منهاج الملقى بين BIOS وعمليات

MSDOS.SYS . ويسمح لهذه الأخيرة بأن تكـون مستقلة عن مواصفـات العتـاد . ويناسب أكثر عمليات LIOCS كما حددناها في الفصل 3 . يحتوي السجل MSDOS.SYS على العمليات التي تؤمن تنظيم السجلات . هذه الأخيرة تُنظِّم الموارد المنطقية من خلال القنوات .

يناسب السجل COMMAND.COM مفسِّر الأوامر ويحتوي على العمليات التي تؤمن الملقى مع المستعمل . ويحتوي على قسمين ، الأول راكن ويقع في الذاكرة بعد عمليات MSDOS.SYS ، ويشحن القسم الآخر أبعد منه ويُمكن أن يُدهس بواسطة برامج المستعملين خلال التنفيذ .

مخطط توزيع مركبات النظام MS-DOS في الذاكرة الحية مصوَّر على الشكل 1.6 .

3 . السجلات

1.3 . التسجيل الفيزيائي على الاسطوانة

تحتوي الاسطوانة ذات الوجهين والمُنسَّقة بإشراف النظام MS-DOS على 40 مساراً في كل وجه . يُقسَّم كل مسار (pist) إلى 9 قطاعات دائرية (sector) ، وكل قطاع دائري يحتوي على 512 بايتة . يوجد إذاً ما مجموعه 368640 بايتة بتصرفنا . ترقيم القطاعات الدائرية يتم على التوالي على كل وجه . مثلاً على المسار «0» نجد القطاعات من 9 إلى 17 على الوجه الثاني . القطاع الدائري رقم 18 موجود على الوجه 1 المسار رقم 1 الخ .

يستعمل النظام MS-DOS مفهوم وحدة التخصيص لتنظيم السجلات . تتألف وحدة التخصيص من قطاعين دائريين أي 1024 بايتة . يتألف كل سجل من عدد من وحدات التخصيص . عناوين وحدات التخصيص تسجَّل في جدول من وحدات التخصيص (File Allocation table : FAT) . يتألف هذا الجدول من مداخل من 12 بتة (3 أرقام سادس عشرية تسمح بعنونة 4096 وحدة تخصيص) . لعنونة 360K بايتة من الأسطوانة ، يستعمل FAT قطاعين دائريَّين مُخزَّنين على الأسطوانة نفسها . تعريف السجلات يتم بواسطة قائمة ، وهذا يدل ولكل سجل على عنوان وحدة التخصيص الأولى .

ربط وحدات التخصيص في حلقات تؤلف سجلاً ، يتم بواسطة FAT . نحصل إذاً على تنظيم وسيطي بين التنظيم المتسلسل والتنظيم المقطَّع (أنظر الفصل

الرابع) . يمكن للقائمة أن تصف 112 سجلاً ، وتستعمل 32 بايتة لكل سجل . يلزم إذاً 7 قطاعات دائرية للأسطوانة .

تنظيم الأسطوانة يبدو إذاً على الشكل التالي :

- برنامج إطلاق على القطاع الدائري الأول ،
- FAT على القطاعين الدائريين التاليين ،
- نسخ FAT (قطاعين) ،
- قائمة (7 قطاعات دائرية) ،
- سجلات المستعملين .

إذا كان ذلك يتعلّق باسطوانة تحتوي النظام نجد ، بعد القائمة :

- السجل IO.SYS (10 قطاعات دائرية) ،
- السجل MSDOS.SYS (34 قطاعاً دائرياً) ،
- السجل COMMAND.COM (23 قطاعاً دائرياً) .

أي ما مجموعه 75 قطاعاً دائرياً . يبقى 322048 بايتة بتصرفنا لسجلات المستعملين . يعطي هذا الرقم على سبيل المثال ، ويتعلق بالصيغ المستعملة للنظام MS-DOS .

2.3 . تنظيم السجلات

الصيغة الثانية من النظام MS-DOS، تسمح بتنظيم السجلات حسب التركيبة الشجرية . وهذا هو أحد عناصر الاختلاف بين النظامين MS-DOS وCP/M لأن هذا الأخير لا يسمح إلا بحماية تجعل سجلات بقية المستعملين ظاهرة. بلوغ السجل يتم بتحديد مسار البلوغ إضافة إلى إسم السجل . وكما رأينا فإن التركيبة الشجرية تعني وجود جذع هو القائمة الأولية التي تحتوي على عناوين السجلات إضافة إلى عناوين القوائم الأخرى .

يُمثّل السجل في القائمة بواسطة 32 بايتة مشغولة كالتالي :

- مدخل حرّ أو لا (0) ،
- إسم السجل (1 إلى 7) ،
- الإطالة (11 إلى 12) ،
- (13 إلى 21 محجوزة) ،

ـ ساعة وتاريخ آخر تعديل (22 إلى 25) ،
ـ حجم السجل (28 إلى 31) .

الخاصية تحدّد حقوق البلوغ على الشكل التالي :

ـ 00 سجل مبلوغ عند القراءة والكتابة ،
ـ 01 سجل للقراءة فقط ،
ـ 02 سجل مخبّأ (كالسجل IO.SYS)
ـ 04 سجل نظام
ـ 08 تشير إلى أن ذلك يتعلّق باسم الحجم (Volume) وليس باسم السجل ،
ـ 10 تشير إلى أن ذلك يتعلّق بقائمه ثانوية وليس باسم السجل .

القائمة الجذرية يمكن أن تُحدِّد حتى 112 سجلاً أو قائمة ـ ثانوية ، ولكن القوائم الثانوية ليست محدودة .

بلوغ السجل الذي يشكّل ورقة من الشجرية يتم من خلال القائمة الجذرية بواسطة مؤشّرات تدل على القوائم الوسيطة . في المستوى المنطقي، أي من وجهة نظر المستعمل ، مسار البلوغ هذا يحدَّد بواسطة سلسلة من الأسماء مفصولة عن بعضها بواسطة قضبان معكوسة . مثلاً :

A : MARC \ TRAVAIL \ LABORAT \ EQUIPE \ BUDGET.TXT.

هذه التركيبة الشجرية تسمح بتوزيع واضح لوحدات الخزن وتؤمن حماية أكيدة . ولكن هذه التركيبة تجعل البلوغ صعباً إذا كان يجب في كل مرّة تحديد كامل الشجرية من خلال الجذع . يمكن الابتعاد عن هذه الصعوبة باستعمال مفهوم القائمة الجارية الاستعمال . من خلال الجذع ، نتحرّك من قائمة ثانوية إلى قائمة ثانوية أخرى بواسطة الأمر «CHDIR» وسجلات القوائم الجارية يتم بلوغها مباشرة . يوجد أربعة أوامر لتنظيم شجرية السجلات :

MKDIR, RMDIR, TREE, et
و PATH

3.3 . الأسماء المحجوزة

يوجد عدد من أسماء السجلات أو الإطالات المحجوزة لأنها مستعملة من قبل النظام. هذه الأخيرة يتم إنشاؤها بواسطة أوامر من اللائحة المعروضة في الفقرة 4 .

أ ـ الأسماء المحجوزة للوحدات :

- A: وB: أسماء وحدات الاسطوانات المرنة أو الاسطوانات القاسية ،
- AUX: الباب الأول المُتسلسل ،
- CLOCK: الساعة ،
- COMn قنال لا تزامني مع n=1 أو n=2 ،
- CON: قنصلة ،
- LPTn: طابعة متوازية ،
- NUL: وحدة خيالية ،
- PRN: الطابعة الأولى المتوازية ،

ب ـ أسماء السجلات المحجوزة

- %PIPE سجل معطيات مؤقت في أنبوب (أنظر 5.3) ،
- @... يُستعمل بواسطة منقِّح الأربطة
- BADTRACK يحتوي على عناوين وحدات التخصيص غير المستعملة ،
- FILE nnn. CHK وحدات التخصيص المفقودة والمُسترجعة بواسطة CHKDSK
- FILEnnn.REC سجل ناتج عن تنفيذ RECOVER (أنظر 4) ،
- NUL تحويل السجل إلى سجل خيالي ،
- VM.TMP سجل مُنشأ بواسطة مُنقِّح الأربطة عندما لا يمكن للزجلة المشحونة أن تدخل في الذاكرة

ج ـ إطالات أسماء السجلات المحجوزة

- $$$ سجل مؤقت ،
- ASM سجل مصدري بلغة المؤول ،
- BAK سجل خزن ينشئه المُنقِّح ،
- BAS سجل مصدري بلغة بازيك ،
- BAT سجل لائحة الأوامر ،
- BIN سجل ثنائي صورة ،
- COM سجل أوامر ،
- CRF سجل مصدري للمراجعة ،

- EXE سجل قابل للتنفيذ ،
- HEX سجل بالترقيم السادس عشري ASCII يُحوَّل إلى ثنائي بواسطة Debug ،
- LIB سجل مصدري من المكتبة ،
- LST سجل لائحة التأويل ،
- MAP جدول الزرع المنشأ بواسطة مُنقِّح الأربطة ،
- OBJ سجل مستهدف قابل للتنفيذ ومُؤوَّل ،
- REF سجل لائحي مرجعي ،
- REL سجل relocatable file من المؤول ،
- TMP سجل مؤقّت

4.3 . السجلات اللوائحية للأوامر

وتسمح بالتنفيذ الأوتوماتيكي للأوامر المتتالية ، متتالية يمكن أن تتكرَّر مع توقف ناتج عن شرط محدَّد مسبقاً أو بنتيجة تدخل المؤثر . تدعى هذه السجلات BATCH في أبجدية MS-DOS وتتميَّز بالإطالة «BAT» . في نهاية كل عملية « إعداد » لنظام التشغيل يجري تنفيذ السجل «AUTOEXEC.BAT» بالتزامن إذا كان موجوداً مما يسمح بشخصنة عملية الإعداد هذه .

بالإمكان إدخال متغيرات شكلية في سجل لائحة الأوامر ، وتُستبدل بواسطة القِيم المحدَّدة عند طلب تنفيذ السجل . هذه المتغيرات تتمتع بأسماء مُحدَّدة وهي من 1% إلى 9% . فلنعاود المثل في الفصل الخامس للتنفيذ المتتالي لأحد البرامج . إذا كنا نرغب باستعمال هذه اللائحة من الأوامر مهما يكن إسم البرنامج القابل للتنفيذ ، فإذاً اللائحة «mont fich . bat» ستأخذ الشكل التالي :

```
ECHO off
:debut
%1
ECHO                          ctrl-c   للخروج يجب ضغط الزر
PAUSE
GOTO :debut                             ( البداية )
```

وسيتمّ نداؤه لتنفيذ البرنامج mon prog على الشكل التالي :

montfich monprog

المُتغيِّـر 1% سيأخذ إذاً القيمة monprog .

إنشاء سجل من الأوامـر يمكن أن يتم بواسـطة المُنقِّـح EDLIN الموجـود مع النظام ، ويمكن إستعمال مُنقحـات أخرى .

هناك خمسة أوامر تسمح بإنشاء تركيبة التحكُّـم البسيـطة التي تُحوِّل مجمـوعة الأوامر من النظام MS-DOS إلى لغة برمجة بسيطة . سنقوم باستعراضها لاحقاً .

- ECHO حيث المُتغيِّـر يمكن أن يكـون «on» ، «off» أو رسالـة ، ويسمح بـرؤيـة الأوامر على الشاشة خلال تنفيذها («on») أو على العكس بـدون عرض أي شيء («off») ؛ عندما يكون المتغير عبارة عن رسالة يجري إرسالها دائمـاً إلى الشاشـة (أنظر التعليمة «write» من لغة البرمجة) .

- FOR وتسمح بتنفيذ متكرِّر مع تزايد قيمة المُتغيِّـر بعد كـل عملية تكـرار ، مثلاً : اللائحة «sauve.bat» التالية :

```
a:
for %%f in (%1 %2 %3) do erase %%f.bak
for %%f in (%1 %2 %3) do rename %%f.txt %%f.bak
for %%f in (%1 %2 %3) do copy b:%%f.txt a:
dir
```

وتسمح باستيفاء يومي لأسطوانة الخزن الموضوعة في الوحدة :a ، باعتبار حتى ثلاثة سجلات في أمرٍ واحد :

sauve toto tutu titi

هكـذا فـالأمـر من السـطر 2 سيمحـو على التـوالي السجـلات a:toto.bak ، a:tutu.bak ، a:titi.bak ، أمّا أوامـــر الســطر 3 فستعيـد تسميـة الـسجـلات : a:toto.txt ، a:tutu.txt ، a:titi.txt لإضافة الإطالة «bak» ، وفي النهاية أمر « السطر الـرابـع » سينسـخ السجـلات b:toto.txt ، b:tutu.txt ، b:titi.txt على إسـطوانـة الخزن :a .

- IF تسمح بتنفيذ أحد الأوامر بشرط معيّـن . والنسق هو التالي :

IF(NOT) commande condition

Commande condition : أمر شرطي .

يوجد ثلاثة أنواع من الشروط :

ـ EXIST nomfichier (حقيقة إذا كان السجل موجوداً) ،
ـ chaine 1 = = chaine 2 (إذا كانت سلسلتا السمات متشابهة) ،
ـ ERRORLEVEL numéro (حقيقة إذا كان البرنامج المُنفَّذ سابقاً مزوّداً بكود خروج يعادل أو أكبر من الرقم المحدَّد) . مثلاً اللائحة desc.bat

```
echo off
if not exist %1.* echo %1
if exist %1.* dir %1.*
```

يسمح بواسطة الأمر «desc.toto» بمعرفة ما إذا كان السجل toto موجوداً مهما تكن إطالته ، إذا كان على الأقل هكذا سجل موجوداً على الاسطوانة الموضوعة قيد الاستعمال ، فسيتم عرض خصائصه على لائحة .

ـ (وسمة) GOTO étiquette: هو أمر للعودة إلى السطر المُحدَّد بواسطة الوسمة étiquette . الوسمة هي عبارة عن سطر من ثماني سمات على الأكثر متبوعة بنقطتين . اللائحة «monfich.bat» المذكورة أعلاه تشير إلى هذا الأمر .

ـ SHIFT عبارة عن أمر يسمح بإضافة متغيرات شكلية بالنسبة للمتغيرات 1% إلى 9% .

3.5 . الأنابيب والمُصفِّيات (tubes and filters)

يعالج النظام MS-DOS في المستوى المنطقي المداخل والمخارج الفيزيائية كسجلات ، ومن الممكن إعادة توجيه المداخل ـ المخارج النموذجية إلى لوحة المفاتيح (Keyboard)القنصلة وشاشتها. هكذا يمكن استعمال الأوامر والبرامج التي ترسل المعلومات الى الشاشة للكتابة على السجل . الأمر «DIR» يعرض لائحة بأسماء السجلات في قائمة معينة على الشاشة ، والأمر :

DIR > mon répertoire

سيعرض القائمة على السجل «monrépertoire» . الرمز «>» يسمح بتوجيه المدخل .

MS-DOS يسمح أيضاً بتعليق البرامج فيما بينها بشكل تصبح فيه المخارج النموذجية (على الشاشة) لأحد البرامج عبارة عن مداخل نموذجية (لوحة مفاتيح القنصلة) لبرنامج آخر . فالأمر :

DIR | MORE

يناسب إذاً إعادة توجيه مخارج DIR نحو مداخل MORE (عرض السجل صفحة بعد صفحة) من خلال السجل المؤقت الموضوع أمام المستعمل والمدعو قنالا . هذا المفهوم هو قريب من مفهوم tube (الأنبوب) في النظام UNIX ، والاختلاف الرئيسي بينهما هو أنه تحت إشراف UNIX يمكن إطلاق مهمتين بالتزامن (واحدة لـ DIR وأخرى لـ MORE) بينما بالنظام MS-DOS يمكن تنفيذ المهمتين على التوالي . وفي النهاية يقدِّم MS-DOS أيضاً برامج مُساعِدة تدعى مُصفِّيات (filters) مستوحاة أيضاً من البرامج المُساعِدة الأكثر إستعمالاً في النظام UNIX . وتتعلَّق بـ :

- FIND الذي يبحث عن سجل موجود حيث الاسم يحتوي على سلسلة من السمات تدعى متغيرات ،
- MORE يعرض مضمون السجل صفحة بعد صفحة ،
- SORT الذي يفرز بالترتيب الأبجدي لوائح أسماء المعطيات في أحد السجلات ، مثلاً :

DIR I SORT

يعطي لائحة بالسجلات الموجودة في قائمة معينة حسب الترتيب الأبجدي .

4 . الأوامر

لائحة الأوامر في النظام MS-DOS هي طويلة . سنعرض هنا أهم الأوامر الرئيسية على سبيل المقارنة فقط مع أوامر النظام CP/M . الشكل المفصَّل ومجموعة إمكانيات إستعمال كل أمر هو موجود في مُساعد المستعمل الذي تقدمه الشركة مع المكنة العاملة بالنظام MS-DOS . أوامر النظام MS-DOS مصنفة داخلية (I) عندما يتم تنفيذها مباشرة بعد شحن النظام في الذاكرة ، وخارجية (E) عندما تتطلب وجود سجلات قابلة للتنفيذ بنفس الاسم على وحدة الاسطوانات العامة .

لائحة الأوامر :

ASSIGN E يوجه جميع الأوامر الموجَّهة لوحدة معينة نحو الأخرى .

BACKUP E يُخزِّن سجلاً أو عدة سجلّات من الأسطوانة القاسية على الاسطوانات المرنة .

BREAK I يجعل العملية CTRL-C صالحة أو غير صالحة .

CHDIR I يُعدِّل القائمة الجارية .

CHKDSK E يُحلِّل قائمة الأسطوانة ويتحقق من تجانسها .

CLS I يمحو الشاشة

COMP E يقارن مضمون السجلات .

COPY I ينسخ سجلاً أو عدة سجلات محدَّدة .

CTTY I يغيِّـر جهاز الإدخال والإخراج .

DATE I يعرض ويُثبِّـت التاريخ

DEL I يصفِّر السجل أو السجلات المحدَّدة .

DIR I يعرض مداخل القائمة المطلوبة .

DISK COMP E يقارن مضمون إسطوانتين .

DISK COPY E ينسخ بالكامل أسطوانة على أخرى.

ECHO I يجعل فعالاً أو يُوقف العرض «ECHO» للأوامر في لائحة من الأوامر .

ERASE I شبيه بـ DEL

EXE2BIN E يُحوِّل مضمون السجلات إلى النظام الثنائي .

EXIT I يخرج من النظام MS-DOS .

FDISK E يقوم بإعداد أقسام MS-DOS للاسطوانة القاسية .

FIND E يبحث عن سلسلة من السمات بداخل واحد أو عدة سجلات .

FOR I يسمح بتنفيذ متكرِّر لأوامر MS-DOS .

FORMAT E يقوم بتنسيق الاسطوانة .

GOTO I يقفز نحو وسمة جديدة في لائحة من الأوامر .

GRAPHICS E يسمح بطباعة الرسوم

HEXDUMP E يعرض مضمون السجل في النظام السادس عشري .

IF I يقوم بالتنفيذ المشروط لأحد الأوامر في لائحة من الأوامر .

MKDIR I يقوم بإنشاء قائمة .

MODE E يجعل طريقة العرض والارسال المتسلسل فعالة .

MORE E يعرض مضمون السجل صفحة بعد صفحة .

PATH I يجعل مسار البحث عن الأوامر فعالاً .

PAUSE I يقطع مؤقتاً تنفيذ لائحة من الأوامر .

PRINT E يضع في سجل الانتظار السجلات للطباعة .

PROMPT I يُعدِّل الرسالة المُرسلة من النظام MS-DOS والتي يُعلِم فيها بأنه جاهـز لتفسير الأمر .

RECOVER E يُرقّـم السجلات الموجودة على بلوكات مضروبة .

REM I يعرض الملاحظات في لائحة من الأوامر .

RENAME I يعيد تسمية أحد السجلات .

RESTORE E يُخزِّن واحداً أو عدة سجلات من الاسطوانة المرنة على الاسـطوانات القاسية .

RMDIR I يلغي قائمة .

SET I يُخصِّص قيمة (سلسلة سمات) لمُتغيِّـر في لائحة من الأوامر .

SHIFT I يُزيد عدد المتغيرات القابلة للاستبدال في لائحة أوامر .

SORT E يربط مدخل ويقوم بإجراء فرز أبجدي .

TIME I يعرض الساعة ويسمح بالتعديل .

TREE I يعرض جميع القوائم ومسارات الوحدة .

SYS E ينسخ سجلات النظام MS-DOS على الاسطوانة المحدَّدة .

TYPE I يعرض مضمون السجل VER I ويعرض رقم صيغة النظام MS-DOS .

VERIFY I يجعل المفتاح «verify» فعالاً أو غير فعـال ويؤدي إلى التحقق من عدم وجود أية فدرة معطوبة .

VOL I يعرض إسم الحجم VOLUME .

5 . التشكيلة (Configuration)

عندما يجري إعداد النظام MS-DOS ، وتغذية الحاسب بـالطاقـة ، يقوم هـذا الأخير بالبحث عن السجل CONFIG.SYS إذا كان موجوداً قبل البدء بتنفيـذ لسجل الأوامر AUTOEXEC.BAT . من الممكن تعديل بعض مواصفات النظام MS-DOS بإدخال الأوامر المعتمدة في السجل CONFIG.SYS .

بالإمكان تشكيل العناصر التالية :

- صحّـة أو عدم صحّـة العملية CTRL-C (BREAK) ،
- عدد ذاكرات الداريء (Buffers) المستعملة في بلوغ الاسطوانات ،
- عدد السجلات الممكن فتحها على التوازي (القنوات) ،
- شحن المعالج بشكلٍ عام بالأوامر ،
- شحن مناهج تنظيم الوحدات الإضافية .

لهذا يمكن إستعمال عدد من الأوامر . مثلاً :

ـ BREAK لفرض على النظام MS-DOS اختبار السمة CTRL-C عند كل تنفيذ لأحد الأوامر أو عند إجراء بعض العمليات الخاصة .

ـ BUFFERS لتحديد عدد ذاكرات « الداريء الجاهزة » (بين 1 و99) ، وهذا العدد هو 5 (نحو النقص) وكل داريء يشغل مساحة تعادل 512 بايتة من الذاكرة ،

ـ DEVICE = LOGUNITE حيث LOGUNITE تحتوي على برامج تنظيم الأجهزة المحيطية غير النموذجية المُضافة .

ـ = FILES عدد لتحديد العدد الأقصى (بين 1 و99) للسجلات الممكن فتحها على التوازي (القنوات) ، نحو النقص هذا العدد يعادل 8 .

ـ SHELL = PROCOM.COM لشحن مفسِّر للأوامر (أو « معالج » للأوامر) مكتوب بواسطة المُستعمل ومخزّن على السجل PROCOM.COM ، هذا المفسِّر سيتم إستبداله بمفسِّر النظام الموجود في السجل COMMAND.COM .

6 . البرامج المُساعدة (UTILITIES)

يصطحب النظام MS-DOS عدد من البرامج المُساعدة وتحديداً :

ـ مُنقِّح السجل (EDIT) ،
ـ منقِّح الأسطر (EDLIN) ،
ـ منقِّح الأربطة (MS-LINK) ،
ـ برنامج التصحيح (DEBUG) .

الفصل الثامن

النظام IRMX 86

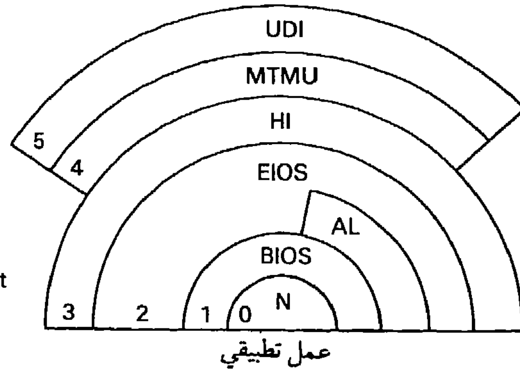
1 . مدخل

النظام iRMX 86 هو نظام تشغيل متعدِّد المهام ومتعدِّد المستعملين موجه للعمل في الوقت الفعلي متطور بواسطة INTEL . وهـو مخصَّص للعمل على عتـاد مجهَّـز بـالـعـائـلة 8086 (iAPX86 ، iAPX88 ، iAPX186 ، iAPX188 ، iAPX286 ، الخ) . هدف هذا النظام هـو تقديم محيط منهجي وبـرامج متنـاسبة مـع التطبيقـات (مثلاً : التحكم بالأعمـال المتواصلة ، أتمتـة العمليات غيـر المتواصلة ، تـطبيقات طبية ، إرسال المعطيات . .) التي يكون فيها الوقت والموارد عبـارة عن مواصفـات حرجة ومهمة .

تركيبة النظام iRMX86 هي عبارة عن تركيبة بـطبقات (أو مستـويات) تسمـح للمستعمل بالعمل على مكنة فرضية معقَّـدة حسب حاجات العمل التطبيقي (شكـل 1.8) .

المستوى 0 هو النواة (nucleus) ويُدعى أيضاً « مرقاب وقت فعلي » (real time executive) . ويُنظِّـم المهام (التي تدعى غالباً عمليات) ، الذاكرة والقواطع .

المستوى 1 والمستوى 2 هما عبارة عن مداخل ـ مخارِج تسمح بجعل المناهج التطبيقية غير مستقلة عن الوحدات الفيزيائية. في المستوى 1 نجـد مُنظِّـم الـوحدات (handlers) ومنظِّـم السجلات حيث المهام تقوم بعمليات نداء لا تزامنية (أي أنها تستطيع متابعة التنفيذ دون توقف) لمهام خدمة BIOS (Basic input output system) .



شكل 1.8 : طبقات النظام iRMX 86

في المستوى 2 تنظيم النداءات هو تزامني (سد المهام المنادية) والبلوغ الى السجلات يتم بواسطة أسماء منطقية . التزويد بذاكرات الداريء وتزامن الإدخال ـ الإخراج هما أوتوماتيكيان . هذا المستوى هو EIOS (نظام إدخال ـ إخراج موسّع) ويناسب بشكل عام الطبقة LIOCS كما حددناها في القسم الأول . مجموعة هذه الأوالية تناسب ما هو مكتوب في الفصل الثالث .

المستوى 3 هو ملقى المستعمل (HI: Human interface) الذي يعطي المستعمل إمكانية بلوغ سهلة للخدمات التي تبذلها الطبقات الدنيا .

من الممكن أن نضيف إلى هذه المستويات مهمتين خاصتين غير متكاملتين فعلياً مع مستوى معين :

- برنامج الإطلاق (bootstrap loader) الوسيط بين العتاد والنواة لأنه موجود في ROM ويستخدم عند إعداد نظام التشغيل .
- شاحن البرامج التطبيقية (application loader) الذي يتعلّق بالنواة (إنشاء المهمة) وبالمستوى الثاني لأن البرنامج المطلوب شحنه موجود على الاسطوانة .

فوق ملقى المستعمل ، يمكن أن نجد مستوى رابعاً لتنظيم متعدد للأدوات الطرفية ومتعدد ـ المستعملين وفوقه أيضاً نجد ملقى نموذجياً للتطوير (UDI) ويُقدِّم لمُصمِّم النظام خدمات مُساعدة في التطوير لتشكيل ، تصريف ، تقويم (déboguage) . . . .

ضمن جميع أنظمة التشغيل المشروحة في هذا الكتاب CP/M ، MS-DOS ،

UNIX) iRMX86 هـو المنـاسب الأفضل لتركيبة .الطبقـات . هذه التركيبة تسمح للنظام iRMX86 بالتكيُّـف مع نوع العمـل التطبيقي : من الممكن العمـل قريبـاً من العتاد بالنسبة للتطبيقات التي تمتاز بمدة جواب قصيـرة جداً ، والتي تهمـل بالكـامل العتاد الذي يعمل باللغات ذات المستوى العال . فهو إذاً إنتاج مُخصَّص للاستعمال من النوع «OEM» (Original equipement manufacturer) .

هنـاك هدف آخـر للنظام iRMX 86 هـو الإلتقاء مـع المنهاج «OPEN NET» للميكروبروسسور INTEL ، والذي يسمح بالإتصال بالمحطات على شبكة مركزية . OPEN NET هو عبارة عن نظام مفتوح يحتوي على سبع طبقات من البروتوكول OSI وISO .

وعلى العكس يجب الاشارة إلى بعض الصلابة فيما يتعلّـق بتنظيم الذاكرة في الصيغة متعددة المستعملين . هكذا فتخصيص الذاكرة لكل مستعمل هو عمل ساكن ويُحدَّد عند التشكيـل . إضافـة لذلـك فإن iRMX 86 ، لا يستعمـل مفهوم الـذاكرة الفرضية .

2 . المرقاب « الوقت الفعلي » (Real time monitor)

يقوم مرقاب الوقت الفعلي بتنظيم المهام ، الذاكرة وعمليات الانقطاع .

1.2 . تنظيم المهام

يرتكز كامل عمل النواة على معالجة إنشاءات المعطيات المعتبرة كمـواضيع . تناسب المواضيع (object) المُحدَّدة في iRMX86 الأنواع التاليـة : أعمال (Jobs) ، مهام (tasks) ، قطاعات (regions) ، Semaphores ، علب رسائل (mail boxes) ، قطع (segments) ، قوائم مواضيع (object directories) ، إطالة (extension) ، مركبة (composite) .

كي يتم التعرّف على الموضوع وكي تأخذه النواة بعين الاعتبار ، يجب أن يتم إنشاء كلّ موضوع بواسطة أحد أصول النواة . عند الإنشاء ، تخصِّص النواة للموضوع معرِّفاً خاصّـاً (object number) يُدعى فيشة (token). الاستعمال الداخلي للموضوع سيتم دائمـاً بـواسطة هذه الفيشة . قيمة الفيشة هي عبارة عن عنوان (20 بتة أو 16 بتة إضافة إلى 4 بتات 0000 offset) الحيز من الذاكرة الذي يحتوي على :

- إما واصف الموضوع إذا كانت تركيبته معقدة (هذه هي حـالة الأعمـال ، المهام ، علب الرسائل ، القطاعات ، المركبات ، Semaphores) .

ـ إما الموضوع نفسه إذا كانت تركيبته بسيطة (هذه هي الحالة مع المواضيع من نوع إطالات وقواطع) .

1.1.2 . المهام

بين المواضيع المحددة في النظام iRMX86 ، فقط المهام هي عبارة عن مواضيع فعالة . لكلّ مهمة بشكل عام عملية خاصة ودورها هو تنفيذ تعليمات الوحدة المركزية وإجراء نداءات لنظام التشغيل . هذا المفهوم يناسب المفهوم المحدَّد في الفصل الثاني ، نفس الشيء بالنسبة للمرقاب الذي يناسب جيداً النواة المحدَّدة سابقاً .

تتميز كل مهمة بعدد من العناصر التالية :

ـ بفيشة التعريف المعطاة لها عند الإنشاء .

ـ بالأولوية المحدَّدة بين 0 و255 ، الأولوية 0 هي الأكبر ، والمهمة الجاهزة ذات الأولوية الأكبر هي الأكثر فعالية .

ـ بالكود القابل للتنفيذ حيث التركيبة هي تركيبة الحلقة غير المُحددة والتي يمكن أن تحتوي على إجراءات قابلة للقسمة بين عدة مهام .

ـ بحالته التي يمكن أن تكون إحدى الحالات التالية : فعالة (runing) ، جاهزة (ready) ، غافلة (asleep) ، معلقة (suspended) ، معلَّقة ـ غافلة (asleep-suspended) ،

ـ بالمتغيرات الأساسية : مضمون مراصف الوحدة المركزية .

ـ إسم المهمة التي أنشأتها في نص « العمل » (Job) .

2.1.2 . حالات المهمة

مختلف حالات المهمة والانتقال بين هذه الحالات هي ممثلة على الشكل 2.8 . ومعناها هو التالي :

ـ المهمة الفعالة : وهي في طور التنفيذ بواسطة المُعالج (أنظر الفصل الثاني) ،

ـ المهمة الجاهزة : وتنتظر المعالج كي يقوم بتنفيذها .

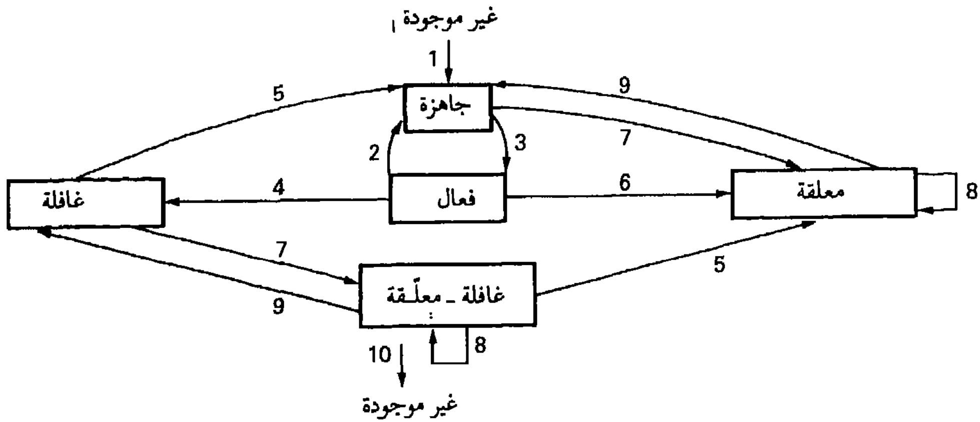
ـ المهمة الغافلة : وهي في الإنتظار (الفصل الثاني) حادثة من الحوادث التالية :

ـ نهاية المدة

ـ إستلام رسالة في علبة من الرسائل

- إستلام إشارة من جهة Semaphore
- إستلام فيشة الموضوع
- السماح ببلوغ القطاع الدائري .

- المهمة العالقة : وهي بانتظار معاودة العمل ، أما تعليق عملها فقد يتم إما بواسطة المهمة نفسها ، وإما بواسطة مهمة أخرى ؛ تتم إعادة الفعالية إما بواسطة مهمة أخرى ، وإما بواسطة إنقطاع (هذا المفهوم هو أقرب من مفهوم المهمة الموضوعة في الانتظار أكثر منه من مفهوم المهمة العالقة كما حددناها في الفصل الثاني) .
- المهمة الغافلة المعلقة : المهمة في الحالة الغافلة يمكن أن تكون أيضاً معلقة بواسطة مهمة أخرى . كي تصبح جاهزة من جديد يجب إيقاظها (نهاية حالة التعليق) وان تستقبل حادثة مُنتظرة (نهايا الاغفاء) وهذا يمكن أن يتم بأي ترتيب ممكن .
- المهمة غير الموجودة : لا يمكن اعتبار هذه الحالة كحالة حقيقية للمهمة ، ولا تعرفها النواة ؛ وسيتم إنشاؤها بواسطة شاحن العمل التطبيقي أو بواسطة مهمة المستعمل .



شكل 2.8 : حالات المهمة

عمليات الانتقال على الشكل 2.8 لها المعنى التالي :

- 1 : تُنشأ المهمة بواسطة (الأصل RQ$CREATE$TASK)
- 2 : تصبح المهمة فعالة لأنها لا تحتوي على أولوية أكبر (الدالة TRANSFERE من الفصل الثاني) .

ـ 3 : تُعلَّـق المهمة مؤقتاً لصالح مهمة بأولوية أكبر (مثلاً : الانقطاع أدّى إلى إيقاظ مهمة أخرى بأولوية أكبر) .

ـ 4 : تنفّذ المهمة الأصل RQ$SLEEP (delay) ، أو توضع في الانتظار لحادثة معينة (علبة الرسائل ، Semaphore ، قطاع . .) .

ـ 5 : إنتاج الحادثة المنتظرة .

ـ 6 : تُعلَّـق المهمة بنفسها (الأصل RQ$SUSPEND$TASK)أو تُوضع في الانتظار لانقطاع معين (الأصل RQ$WAIT$INTERRUPT) .

ـ 7 : تعلّق المهمّة بواسطة مهمة أخرى (الأصل RQ$SUSPEND$TASK)

ـ 8 : تعلّـق المهمّـة بواسطة مهمّـة أخرى وتبقى معلّـقة ، ودرجة تعليقها تزيد واحداً أو تستقبل المهمّـة إشـارة إيقاظ (الأصل RQ$SUSPEND$TASK) ولكن تبقى معلّقة لأن درجة تعليقها تبقى أكبر من صفر بعد أن يتم تنقيصها « واحداً » .

ـ 9 : المهمة هي مُتيقّـظة وتصبح جاهزة أو غافلة .

ـ 10 : يتم القضـــاء على المهمـــة (RQ $ DELETE $TASK) وتُلغى من جميـــع اللوائح ، يُمكن أن نبلغ هذه المهمة من خلال حالة معينة .

3.1.2 . الأعمال (JOBS)

يُحدِّد العمل المحيط حيث تعيش مجموعة من المهام . يحتوي العمل بشكل عام على جميع الأعمال المناسبة لعمل تطبيقي معين كالمهام ، علب الرسائل الخ ، بهذه الطريقة يمكن وضع مجموعة من المهام في العمل بشكل إنشائي (عمل مختلف لكل عمل تطبيقي) . هكذا ، فالإتصال بين مهمتين موجودتين في عمليتين مختلفتين هو ممكن لأنّ مجموعة الأعمال هي منظمة بالتركيبة الشجريـة ومن الممكن دائماً أن تكون في عمل (قريب من الجذع) يحتوي على مهمتين .

عند إعداد نظام التشغيل يتمّ إنشاء عمل يُدعى « جذع » (root job) (في كـل مرَّة يتم فيها إنشاء عمل معيّن يجري إنشاء عمل أوّلي بشكل متزامن). حسب مخطط النظام المهمة المرتبطة بالعمل « جذع » ستنشأ مجموعة من الأعمال (مثل عمل لكل قنصلة) حسب متغيرات محدَّدة عند التشكيل . بـالنتيجة ، يمكن للعمـل أن ينشيء ديناميكياً أعمالاً أخرى .

يتمتع كل عمل (job) بمجموعة من المواضيع (مهام ، علب رسائل ، قطع ، semaphore) التي يمكن أن تكون مبلوغة من خلال قائمة معينة ، إضافة إلى pool في

الذاكرة الحيّة . عندما يقوم العمل بإنشاء أعمال من نوع «fils» (أبناء) فليس بإمكانه تخصيص pool من الذاكرة إلا من أقسام pool الخاصة به والتي لا تعود تنتمي إليه . يُحدِّد pool بواسطة قيمة دنيا وعليا مما يسمح ببعض الديناميكية في مستوى تنظيم الذاكرة . عند إنشاء العمل يجري تخصيص القيمة الدنيا لـ pool له وعند الحاجة بإمكانه إعارة الذاكرة الى أبيه في العمل ولأسباب حماية تُخصَّص الأعمال عند الانشاء بذاكرة pool ساكنة .

4.1.2 . Sémahpores

هي عبارة عن مواضيع تستعمل للتحكُّم بالموارد بين المهام المُتعاونة . وتسمح بتأمين تبادل الموارد فيما بينها . يُنشأ الـ Semaphore بـواسطة الأصـل -RQ $CRE ATE $ SEMAPHORE ويُلغى بواسطة RQ$DELETE$SEMAPHORE .

تُناسب SEMAPHORE المُحدَّدة في iRMX86 المفهـوم المُحدّد في الفصل 2 . يحتوي Semaphore على عداد (counter) مُعالج بواسطة RQ$ RECEIVE $ UNITS التي تناسب «P»(بمدة أطول من الانتظار) وRQ$SEND$UNITS التي تناسب «V» . وتحتوي أيضاً على لائحة بـالمهام في الانتـظار التي تُنظَّم بـواسطة الدورة (على طريقة الذي يصل أولاً يُعالج أولاً) FIFO .

5.1.2 . القطاعات

القـطاعات هي عبـارة عن مواضيـع دورها تحـديد البلوغ للقـطاعات الحـرجة للأكواد والمعطيات . هذا هو مفهوم قريب من Semaphore ، والفـرق الأساسي هـو أنّـه عندما تستلم المهمة الأذن بالدخول إلى قطاع حرِج محمي بواسطة قطاع معين لا يمكن أن يتم تعليقهـا والقضـاء عليهـا من قِبل مهمـة أخرى ، وهـذا ما يُخفِّف من أخطار الوقف المُميتة .

ينشأ من الموضـوع « قطاع » بـواسطة الأصـل RQ $CREATE $REGION ، وطلب البلوغ يتم بواسطة RQ$ACCEPT$CONTROL إذا لم نرغب بالانتظار في حـالة إنشغـال القطاع أو بـواسـطة RQ$RECEIVE$ CONTROL ، الخـروج من القطاع يتم بواسطة RQ$SEND$CONTROL .

6.1.2 . علب الرسائل

علب الرسائل هي عبارة عن مواضيع موجّـهة للسماح بالتعاون بين المهام بينما تقوم semaphore والقطاعات بالتحكـم بالتنافس بين المهام لبلوغ الموارد المشتركة .

تسمح علبة الرسائل بتبادل المواضيع بين المهام . ويمكن أن تتعلّـق بأي نوع

من المواضيع ، ولكن بشكل عام هي عبارة عن معطيات تتبادل بها المهام ، والمواضيع المتبادلة هي عبارة عن قطع (segments) تحتوي على هذه المعطيات . هكذا فالمواضيع ليست قابلة للتبادل فيزيائياً (منسوخة) ، ولكن يجري تبادل الفيش فقط بين المهام وبواسطة علبة الرسائل .

تبادل الموضوع يتم بواسطة الأصل RQ$SEND$ MESSAGE ، الاستقبال ، أي طلب موضوع في علبة رسائل ، يتم بواسطة RQ$RECEIVE$ MESSAGE (مع إنتظار محدود أو غير محدود) . هذه الأصول تناسب مفهوم الإرسال والاستقبال المحدَّد في الفصل الثاني . يرتبط بكل موضوع « علبة رسائل » لائحتا انتظار (في كل لحظة تكون إحدى اللائحتين فارغة) .

- سجل المهام بانتظار « موضوع » .
- سجل المواضيع ينتظر إستهلاكه من قِبل المهام .

تنظيم هذه السجلات هو من النوع « من يصل أوّلاً ، يُلبَّى أوّلاً » (FIFO) .

7.1.2 . قوائم المواضيع

تُستعمل قوائم المواضيع لبعض المهام التي ترغب بجعل بعضٍ من مواضيعها مبلوغة . لذلك فهي تنشر المواضيع وتضع فيشها في قائمة وأسماءها في قائمة « عمل » معيّن (job) . جميع المهام الداخلة في عمل معين يمكن أن تبلغ فيش المواضيع المفهرسة في قائمة في هذا العمل باعطاء إسم الموضوع فقط . مهام الأعمال الأخرى يجب أن تُعطي إسم موضوع وفيشة العمل حيث توجد القائمة التي تحتوي على المعلومات . المواضيع المفهرسة في القائمة « عمل جذري » (root job) هي إذاً مبلوغة لجميع المهام .

8.1.2 . المواضيع من نوع إطالة ومركّب

يعطي نظام التشغيل iRMX86 إمكانية إنشاء مواضيع خاصة للمستعملين بواسطة الأصل RQ$CREATE$ EXTENSON وإلغائها بواسطة RQ$ DELETE $EXTENSION.

من الممكن أيضاً تركيب أنواع من المواضيع بواسطة RQ$CREATE $COMPOSITE وتعديلها بواسطة RQ$ ALTER $COMPOSITE .

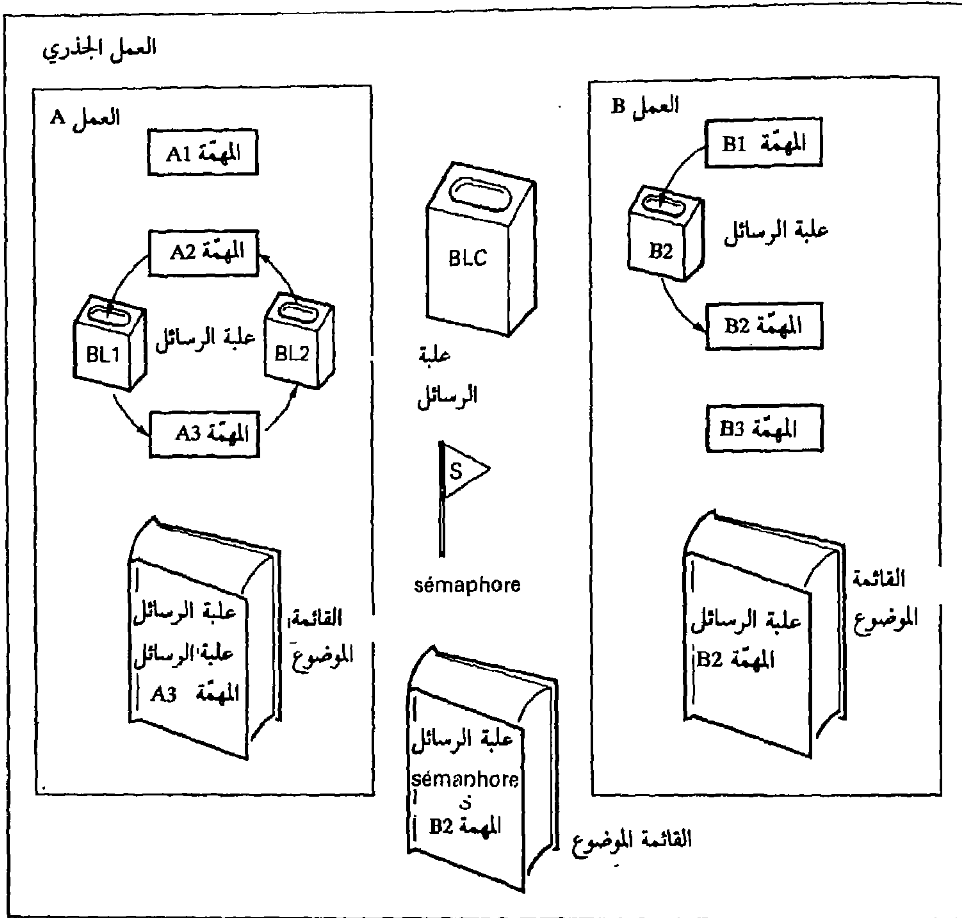
9.1.2 . خلاصة

نرى ان نظام التشغيل iRMX86 يسمح بتركيبة إنشائية جيدة للتطبيقات بتنظيم

المواضيع المستعملة بشكل واضح . الشكل 3.8 يعطي مثلاً بسيطاً على هكذا إنشاء .

علب الرسائل BL1, BL2 يمكن أن تستخدم لنقل المواضيع بين المهام A1 وA2 لأن فيشها مبلوغة من قائمة العمل (job) A .

الـ S Semaphore هو مبلوغ لجميع مهام النظام لأنه محدَّد في قائمة العمل الجذري (root job) .



شكل 3.8 : مثل على إستعمال المواضيع

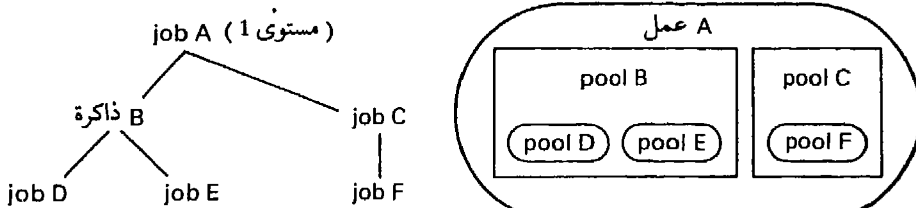
2.2 . تنظيم الذاكرة

يقوم تنظيم الذاكرة على توزيع ساكن ثمّ ديناميكي للمساحة من الذاكرة المعنونة (1 MEGA BYTE) بين الأعمال (Jobs) وبعد ذلك بين المهام . عندما نقوم بتشكيل النظام الذي يحتوي على عدة قناصل (كي نسمح في النهاية للمستعملين بالعمل بشكل متوازٍ) يتم إجراء توزيع ساكن بين الأعمال من المستوى الأول (عمل من المستوى الأول لكل قنصلة) . نرى أن هذا التوزيع الساكن هو مربوط بالقناصل وليس مباشرة بالمستعملين . تظهر لنا التجربة أن الحيز من 200K بايتة على الأقل هو ضروري لكل مستعمل كي يتمكن من العمل بشكل طبيعي .

بعد ذلك ، عندما يستلم كل عمل من المستوى الأول الـpool الخاص به من الذاكرة ، فإن المهمة الأولية لهذه الأعمال تُمكن أن تُنشىء أعمالاً من نوع « ولد fils » وتُخصِّص لها قسماً من pool الخاص بها . يتم إنشاء كل عمل مع pool من الذاكرة مُحدَّد بواسطة متغيّرين : قيمة دنيا تعادل كمية المعلومات المُخصَّصة للعمل عند إنشائه ، وقيمة قصوى من الذاكرة تناسب حدّاً معيناً لا يمكن للعمل أن يتجاوزه في أي وقت . بين هاتين القيمتين يتم التخصيص حسب الحاجة من خلال ذاكرة pool المُخصَّصة للعمل « أب father » ، نقول إذاً إن الأعمال « أولاد » (fils, sons) إستعاروا ذاكرة من الأب .

من جهة أخرى عندما يتم القضاء على عمل « ولد » بواسطة المهمة المرتبطة بعمل « والده » ، يتم إستعادة الذاكرة pool المُخصَّصة له وإضافتها إلى ذاكرة pool للأب . يوجد إذن تنظيم ديناميكي للذاكرة بين الأعمال . وبشكل خاص فإن مجموعة أعمال « أولاد » عمل معين لا يمكن في أي حالة أن تكون مجهَّزة بذاكرة أكبر من قيمة الذاكرة pool المخصَّصة للأب .

الشكل 4.8 يعطي مثلاً على إدخال الذاكرات pool المناسبة للتركيبة الشجرية للأعمال .



شكل 4.8 : الذاكرات pool

تُقسَّم كل ذاكرة pool إلى قطع (segments) تُستعمل بواسطة المواضيع المنتمية إلى العمل المناسب . القطعة هي عبارة عن مجموعة متجانسة من كلمات الذاكرة (من 16 إلى 64k بايتة) . مفهوم القطعة المُستعمل في iRMX86 لا يتناسب أبداً مع المفاهيم الكلاسيكية لأنظمة التشغيل، المفصَّلة في الفصل 2 ، حيث القطعة مرتبطة منطقياً بالبرنامج .

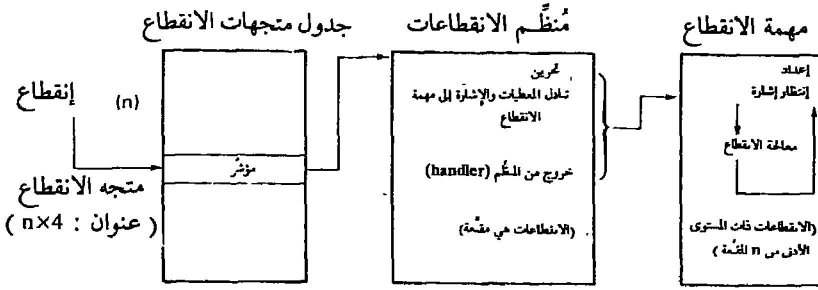
تستعمل القطع لتخزين مكادس (stacks , piles) المهام ، المعطيات ، ذاكرات الداريء (Buffers) المُستعملة في نظام التشغيل ، الكود القابل للتنفيذ للمهمة ، الرسائل المتبادلة بين المهام ، الخ . تنظيم الذاكرة في النظام iRMX86 يتم بمساعدة مفهوم الـ pool والقطع . أوالية الذاكرة الفرضية هي غير موجودة حتى الآن .

3.2 . تنظيم عمليات الانقطاع

إنّه مفهوم مُميَّز في iRMX86 لأن هدفه هو التطبيقات في الوقت الفعلي . مُنظِّم المهام يرد على حوادث (event-driven) ويستعمل تقنية (المهمة الفعالة هي دائماً المهمة ذات الأولوية الكبرى) تسمح بخدمة الحوادث ذات الأولوية العالية قبل الأخرى . النظام iRMX86 يقدِّم مستويين لتنظيم عمليات الانقطاع : منظِّم الانقطاعات (handlers) ومهام الإنقطاع .

منظِّم الانقطاعات يُشكِّل المستوى الأول ، ويرتبط مباشرة بالعتاد ويُطلق في العمل بواسطة إشارات الانقطاع . وهدفه هو تحديد الحادثة وكذلك يجب وقف جميع المهام الأخرى . يجب إذن أن تُقدِّم خدمة في مهلة بسيطة لأن جميع عمليات الانقطاع هي مُقنَّعة خلال التنفيذ . إذا كانت المعالجة بسيطة ، يمكن أن تتم في داخل مُنظِّم لعمليات الانقطاع ، ولكن بشكل عام نحاول إرسال اشارة الى مهمة الانقطاع لانهاء الخدمة المطلوبة وستصبح فعالة حسب أولويتها .

تتمتع مهام الانقطاع بأولوية مثبّتة بواسطة مستوى إنقطاع عتادي معيَّن وهي بشكل عام عبارة عن « مهام مباشرة » بمستوى أولوية أعلى من المهام التي ترد فقط على حوادث ناتجة عن المناهج الداخلية . خلال دوران إحدى مهام الإنقطاع فإن كلّ الإنقطاعات بمستوى أقل أو يساوي المستوى الذي يخدم المهمة هي مقنَّعة . نوجز هذه الأوالية على الشكل 5.8 .



شكل 5.8 : أوالية تنظيم الانقطاعات

3 . تنظيم الادخال ـ الإخراج

يتمّ تنظيم الإدخال ـ الإخراج على مستويين .

- (Basic input output system) BIOS
- (EXTENDED input output system) EIOS

BIOS . 1.3

يُقدِّم BIOS بلوغاً مباشراً لوحدات الإدخال ـ الإخراج . هذه الامكانية هي ميزة أنظمة التشغيل الموجهة نحو الأعمال التطبيقية العاملة في الوقت الفعلي . تقوم المهام التطبيقية بنداءات لا تزامنية للبرنامج BIOS مما يسمح بمتابعة تنفيذها . وإذا كان يجب أن تنتظر حتى إنتهاء إحدى عمليات الإدخال ـ الإخراج فهي تتوقف في انتظار علبة الرسائل، مما يسمح بمتابعة عمليات إدخال ـ إخراج أخرى مع عمليات أخرى .

يتصل البرنامج BIOS مع الوحدات الفيزيائية بواسطة إجراءات نموذجية (de- vice handlers drivers) أو (device handlers) تقوم بأربع عمليات أساسية للتحكُّم بالوحدات والاتصال بها :

- إعداد وحدات الإدخال ـ الإخراج
- نهاية الإدخال ـ الإخراج
- سجل إنتظار للإدخال ـ الإخراج
- إلغاء طلب بالإدخال ـ الإخراج

النظام iRMX86 هو نظام مفتوح بمعنى أن هذه العمليات الأساسية يمكن أن تستخدم لكتابة إجراءات أساسية (device drivers) متكيِّفة مع الوحدات الفيزيائية غير النموذجية .

في الإجراءات النموذجية يدخل التبادل المتوازي بواسطة Universal USART) synchronous receiver transmitter) ، الملقى المتوازيCentronics للطابعة وعدد كبير من الوحدات iSBC ، iSBX من إنتاج (diskette controler, disk winchester) INTEL . إضافة لذلك ، فإن كل إجراء نموذجي (device driver) أساسي يمكن أن يُستعمل كملقى مع مختلف الوحدات الفيزيائية . في الواقع تحتوي على عدة تحكّمات أو مراقِبات للوحدات (controlers) .

EIOS . 2.3

يُضيف البرنامج EIOS إمكانيات إضافية للبرنامج BIOS . ويقوم بتنظيم أوتوماتيكي للداريء ومزامنة طلبات الإدخال ـ الإخراج . التنظيم الأوتوماتيكي لذاكرات الداريء يسمح بالحصول على بلوغ مثالي للوحدات وذلك بالاشتراك في طلبات المهام (تعبئة داريء القراءة للسجل) أو بتفادي توقف المهام (داريء الكتابة) .

يستمل البرنامج EIOS نداءات شبيهة بنداءات البرنامج BIOS الموضوعة جانباً بالتأثير بشكل تزامني وبتعليق أوتوماتيكي للمهام التي تطلب الخدمة حتى الانتهاء من هذه الأخيرة . يسمح EIOS بالعمل مع الوحدات ومع السجلات بواسطة الاسماء ، ويستعمل مفهوم الوحدة المنطقية ويسمح باستقلالية كاملة للمناهج التطبيقية لجهة مميزات العتاد المُستعمل .

4 . تنظيم السجلات

يوجد ثلاثة أنواع من السجلات في النظام iRMX86 : السجلات المسمّاة (named files) ، السجلات الفيزيائية (physical files) وسجلات القنوات (Stream files) . كل نوع يمكن أن يبلغ وحدات الإدخال ـ الإخراج من خلال إجراءات نموذجية .

1.4. السجلات المسمّاة

وتعرّف بواسطة سلسلة من السمات ASCII تُشكّل إسم السجل . تُخزّن هذه الأسماء في قائمة عبارة عن سجل مسمّى أيضاً . من الممكن إذن أن تحتوي قائمة معينة على إسم قائمة أخرى . هكذا فالنظام iRMX86 يسمح بتركيبة شجرية للسجلّات (أو تراتبية hiérarchique) كما هي الحالة في النظامين MS-DOS وUNIX . نبلغ السجلات بواسطة مسارات في الشجرة وذلك عندما نُراجع القائمة

الأساسية ، ولكن بالامكان أن نبلغ السجلات الموجودة في القائمة العاملة مباشرة .

تتميّـز السجلات بأبعادها ، وبحق بلوغها ، وعنوانها الفيزيائي . يعتمد النظام iRMX86 شكلين للسجلات : صغيرة وكبيرة . يؤشر السجل الصغير على 8 فـدرات متواصلة من المعطيـات كحدّ أقصى . إذا كـان يجب فصل المعـطيات لكي تشغـل وحدات الخزن بشكل جيّـد فإن نوع السجل يُحوَّل أوتوماتيكياً إلى سجل كبير . حجم السجلات الكبيرة هو غير محدود ، ما عدا بالنسبة لامكانيات الوحدات الفيزيائية .

هنـاك إمكـانيــة لحمـايــة السجـلات في النــظام iRMX86 . يُمكِن تصنيف المُستعملين في مجموعات بحقوق بلوغ مختلفة لكل مستعمل (user) ، للمُستعملين المُنتمين لنفس المجموعة (group) أو لجميع المستعملين (world)

2.4 . السجلات الفيزيائية

وتُستعمل للبلوغ المباشر للوحدات . يشغـل السجل الفيـزيائي كـامل الـوحدة المربوط فيها . لا يوجد تحكّـم ببلوغ السجلات الفيزيائية .

3.4 . السجلات قنوات (Stream)

وتستخدم لتبادل المعطيات بين المهام مباشرة بواسطة قنال الذاكرة الحيّـة

5 . ملقى المؤشر (Human interface)

هو القسم الخارجي من iRMX86 (Human interface) الذي يسمح بإدخال الأوامـر . ويحتوي على مفسّـر لـلأوامـر يتحقّق من صحـة نحـو الأوامـر وصـلاحيـة المتغيرات قبل إطلاق تنفيذها بواسطة إنشاء عمل (job) . يتم تنفيذ الأوامر واحداً بعد الآخـر كما في CP/M أو MS-DOS ويمكن أن يتم تجميعها في سجّل قابل للتنفيذ مبـاشرة بـواسطة الأمـر SUBMIT . نجد هنـا نفس النوع من الأوامـر كبقية أنـظمـة التشغيل : DIR ، RENAME ، COPY ، TYPE ، . . . الخ .

عندما يتم إستعمال iRMX86 في الصيغة متعدِّدة المستعملين ، يمكن الحصول على الحالة master-user (مُستعمل رئيسي) من خلال أية قنصلة وذلك بضرب الأمر SUPER مصحوباً بكلمة عبور . عند ذلك يكون بإمكاننا بلوغ جميع السجلات مهما تكن درجة حمايتها .

يتمتع النظام iRMX86 بعدد كبير من الأوامـر إن في مستوى نـداءات الأنظمـة (تنظيم المواضيع ، الذاكرة ، الإنقطاعات) أو في مستوى مُلقى المستعمل .

6 . التشكيل

النظام iRMX86 هو نظام مفتوح موجّه نحو العمل في الوقت الفعلي والمُستعملين OEM ، وتشكيله عبارة عن عملية معقدة . هكذا ، لا يكفي فقط تشكيل النظام حسب عدد القناصل ، أو الاسطوانات والطابعات . فالنظام iRMX86 هو قابل للتشكيل في طبقات ، وفي داخل كل طبقة من الممكن تحديد مهام نظام التشغيل التي نرغب باستعمالها بالغاء المهام الأخرى . هذا ما يسمح لنا بالحصول على نظام تشغيل بحجم صغير متكيّف بشكل أفضل مع العتاد والمناهج التطبيقية . لاجراء تشكيل للنظام ، يوجد منهاج تخاطبي للتشكيل يُدعى (interactive configuration utility-ICU من إنتاج INTEL) .

ICU يتمتع بالمراحل الأربع التالية :

- في المرحلة الأولى يجري عرض عدد من الشاشات على قنصلة النظام الأساسية ، وتعرض لنا مجموعة من الخيارات ، اتجاهات نحو النقصان تسمح بتحديد الحالة الأكثر تداولاً .
- في المرحلة الثانية ، يوجد سجلات وصف (بلغة المؤوّل أو باللغة PLM86) يجري عرضها .
- في المرحلة الثالثة ، يقوم ICU بعملية تنقيح للأربطة بين هذه السجلات وبين المكتبات الموجودة في كود قابل للتنفيذ ؛ ينتج عن ذلك زجلة في كود قابل للتنفيذ .
- المرحلة الرابعة تزرع كل طبقة في الذاكرة على الشكل زجلة (module) بعناوين فيزيائية ؛ هذه الزجل يمكن أن يتم نقلها إلى سجل موحّد قابل للشحن سلكياً على المكنة المستهدفة التي نرغب بتشكيلها .

نرى هنا أن عملية التشكيل هي أكثر صعوبة من الأنظمة الكلاسيكية . من الممكن أيضاً تشكيل أنظمة تشغيل مُتخصّصه بدون وحدات خزن خارجية من خلال نظام آخر يعمل بالنظام iRMX86 وذلك بشحن نظام التشغيل سلكياً .

الفصل التاسع

النظام UNIX

1 . مدخل

يغطي الاسم «UNIX» عائلة من أنظمة التشغيل . هكذا ، لا يوجد فقط بعض الصيغ التي نشرتها ATT ولكن يوجد عدّة أنظمة تشغيل متكيّفة مع UNIX ، كالنظام XENIX أو ULTRIX . في البداية ، جرى تطوير UNIX في مختبرات Bell المرتبطة بـ ATT والصيغة الأولى ظهرت في سنة 1970 . حالياً ، إنّه نظام تشغيل ينتشر في الأوساط الجامعية ، إحدى مميزاته هي في « إنفتاحه » أي الإمكانية المعطاة للمستعمل في إنشاء الأصول الخاصة به والأوامر ، وتلك التي تسمح له بالعمل في محيط من العتاد غير متجانس . سيئة هذا الانفتاح تكمن في أنّ أوالية الحماية يمكن إختراقها بسهولة .

النظام UNIX هو نظام متعدِّد المهام ومتعدِّد المستعملين وهو غير مُوجَّه للتحكّم بالعمليات الصناعية في الوقت الفعلي . بعض الصيغ من هذا النظام تحتوي على إطالات تسمح بتفادي ولو جزئياً هذه الثغرة . هكذا ، وفي الأصل ، الميكانيكية الوحيدة للاتصال بين المهام هي عبارة عن الأنبوب (tube) الذي يناسب الاستعمال المقسّم للسجل . تُكتب المهمة الأولى على السجل وتأتي الأخرى لقراءة المعلومات بشكل لا تزامني حسب حاجاتها . هذه الميكانيكية هي فعالة في كل مرّة نحتاج فيها إلى تزامن من نوع « مُنتج ـ مستهلك » وهي مفيدة لتعليق أوامر نظام التشغيل . مثلاً :

ll I more

يسمح بعرض جميع مميزات السجل من القائمة العاملة (الأمر ll) وبعرضها

صفحة بعد صفحة على الشاشة (الأمر more) . على العكس ، هذه الأوالية ليست مُتكيِّفة بشكل جيد لتحديد مخططات معقدة للتبادل والتزامن بين المهام من النوع الذي نلتقيه لتحديد التعاون بين المهام في نظام تحكّم بالعمليات الصناعية .
في هذا المفهوم فإن UNIX لا يُقدِّم سوى الأصلين «Wait» و«Signal» إضافة إلى أواليات إرسال وإستقبال للرسائل في علب رسائل طبيعية وسهلة الاستعمال .

على العكس ، فإن تنظيمه الفعّال للتركيبة الشجرية لنظام السجلات ، والتنظيم الدينامیكي للذاكرة والتقسيم المتساوي للوحدة المركزية بين المهام والمُستعملين يجعل من هذا النظام نظاماً ممتازاً للتشغيل في صيغة متعدِّد المستعملين .

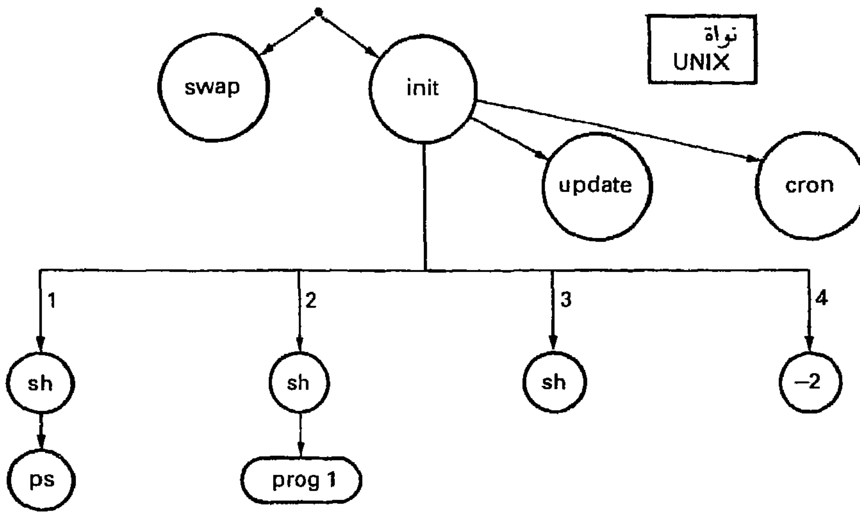
2 . النواة

النواة هي عبارة عن مجموعة من الإجراءات والدوال المكتوبة في القسم الاساسي منها بلغة C . هذه الاجراءات تُستدعى بواسطة المهام ولكن تُنفَّذ في صيغة مُميَّزة (صيغة النواة) عن بقية إجراءات المُستعمل .

حول النواة ، نجد عدداً من المهام الراكنة في الذاكرة الحيّة . أوّلاً نلتقي «swaper» الذي يسمح بأوالية ذهاب وإياب عندما تكون الذاكرة الحيّة غير كافية لمجموعة المهام المُنفّذة (الصيغ الأخيرة من UNIX تسمح بأوالية الذاكرة الفرضية) والتي تقوم بتنظيم الذاكرة . العملية «init» هي عبارة عن مهمة تكمن في إعداد وإنشاء « مهمة » لكل أداة طرفية من النظام ، ومن خلالها يُمكن للمُستعمل أن يفتح دورة عمل «session» (login) . كقسم من المهام الراكنة نجد أيضاً المهام «update» التي تقوم باستيفاء يومي ودوري للسجلات المعدَّلة (بشكل شبيه مع BIOSكما حددناها لأنّ ذلك يتعلّق بنسخ ذاكرات الداريء على الناقل الفيزيائي أو الأسطوانة) و«cron» تسمح بتنفيذ أوامر التاريخ المحدَّدة (إستيفاء يومي للتاريخ والوقت مثلاً) وإجراء تنظيم للوقت بشكل عام .

الشكل 1.9 يعرض هذا التنظيم للمهام في حالة نظام يحتوي على أربع قناصل . تُمثّل المهام الفعّالة بواسطة دوائر . المهام ، المُنشأة بواسطة المهمة «init» والمُخصَّصة لكل أداة طرفية («sh» و«2–») لا تُشكّل قسماً من النظام وهي موضّحة في الفصل الخامس .

كما ذكرنا ، فإن UNIX ليس موجّهاً نحو الأوامر في الوقت الفعلي للعمليات



شكل 1.9 : تنظيم المهام

الصناعية ، ولكن جرى إعتماده لقسمة الوحدة المركزية بين عدة مستعملين بشكل متساوٍ . أوالية المهمة تتطور خلال المدة وليس باستطاعة المستعملين التحكّم بها . مثلاً ، عندما تقوم إحدى المهام بتنفيذ إجراء معين من النواة ، فإن أولويتها تزيد ، على العكس فإن أولوية مهمة تستعمل الوحدة المركزية خلال مدة طويلة تنقص . هذه سيئة بالنسبة لأنظمة التحكّم بالعمليات الصناعية حيث يجب على المُصمِّم أن يتحكّم بالكامل بالأولويات لأن هذه الأخيرة تتغيّر حسب عمل المهام وليس حسب الوقت . على العكس هذا التصرّف هو مفيد في محيط متعدِّد المستعملين .

لنفترض ، مثلاً ، بسبب خطأ ما في البرمجة أن مهمة بأولوية عالية تدور بشكل متواصل في حلقة . ففي نظام متعدِّد المهام حيث المهام مثبتة يمكن أن يؤدي هذا إلى توقّف عام لجميع المستعملين ، والمهمة المغلوطة تسيطر على المعالج . بإشراف UNIX ، أولوية هذه المهمة ستنقص وتنتهي بأن تصبح أقل من أولوية بقية المستعملين . سيكون بإمكان المسؤول عن النظام أن يقوم يدوياً بالقضاء على المهمة الخاطئة دون التأثير على بقية المُستعملين .

3 . المداخل ـ المخارج

وحدات الإدخال ـ الإخراج هي محدَّدة كسجلات خاصة مجموعة في قائمة خاصة تدعى «dev/».تتمتع هذه السجلات بخاصية تعادل «b» إذا كانت الوحدات

تعمل في فدرات (الأسطوانة مثلاً) و«c» إذا كانت الوحدات تعمل بالسمات (أداة طرفية مثلاً) . لا يوجد إذن مُنظِّم إدخال ـ إخراج يُحدِّد بشكل واضح طبقة نظام التشغيل بإشراف UNIX . هذا التنظيم يتم بواسطة مجموعة من إجراءات النواة التي تُطلب مباشرة بواسطة المهام . هذه الإجراءات تعمل باستعمال «pool» من ذاكرات الداريء (buffers) وتدعى «cache» . الإجراء الذي يقوم بنسخ ذاكرات الداريء على الوحدات الفيزيائية يُدعى «sync» . وهو مبلوغ من المؤثر على شكل أمرٍ واضح ، ولكن يُنفَّذ بشكل متناوب وواضح أمام المستعمل بواسطة المهمة «update» .

4 . نظام السجلات

كما بالنسبة لمنظِّم الادخال ـ الإخراج ، فإن مُنظِّم السجلات هو غير موجود كوحدة مستقلة محدَّدة بشكل واضح بإشراف UNIX . يتم إنشاء مهامّه بواسطة مجموعة من إجراءات النواة ومهام الخدمة المنشأة ديناميكياً عند الحاجة .

لقد سبق أن أعطينا النظام UNIX كمثل على الأنظمة المرتكزة على تركيبة شجرية للسجلات . يجب أيضاً إضافة مفهوم تركيب الحجم «volume» . ففي أنظمة مثل CP/M وMS-DOS . نجد مُنظِّم عمليات الإدخال ـ الإخراج لا يُدير في لحظة معينة ، سوى وحدة خزن خارجية (وحدة إسطوانات : A أو :B أو إسطوانة قاسية)، ولكل وحدة يوجد جذع وشجرية خاصة بها. على العكس ، فبإشراف النظام UNIX ، يوجد جذع واحد «root» يناسب الوحدة التي من خلالها تُشحن نواة نظام التشغيل عند الإعداد . من خلال هذا الجذع يؤلف نظام التشغيل شجرية من السجلات مبلوغة في لحظة معينة ، مهما تكن هذه السجلات ، موجودة أو غير موجودة في الحجم (volume) أي على نفس وحدة الإدخال ـ الإخراج . فالنظام هو الذي يُحدِّد الحجم حيث يوجد السجل المعيَّن من خلال مسار البلوغ . الإسم المنطقي للحجم هو إسم السجل الخاص المناسب لوحدة الإدخال ـ الإخراج . لبلوغ السجل غير الموجود على الحجم الأساسي (حيث يوجد الجذع) يجب إجراء عملية تركيب الحجم حيث يوجد هذا السجل . مثلاً تركيب الحجم dev/d333/ يمكن أن يتم بواسطة الأمر التالي :

mount/dev/d 333/m n t

حيث dev/d333/ هو إسم السجل الخاص المناسب لوحدة الاسطوانات 333 وmnt/ هو إسم القائمة من الشجرية الأولية حيث الحجم مُركَّب . جذع الشجرية المناسب للحجم dev/d333/ سيتطابق مع القائمة mnt/ مع تخبئة المضمون القديم وسجلات

الاسطوانة 333 ستكون مبلوغة بالعبور بواسطة هذه القائمة . العملية المعكوسة لتركيب الحجم (mount) هي التفكيك (Umount) .

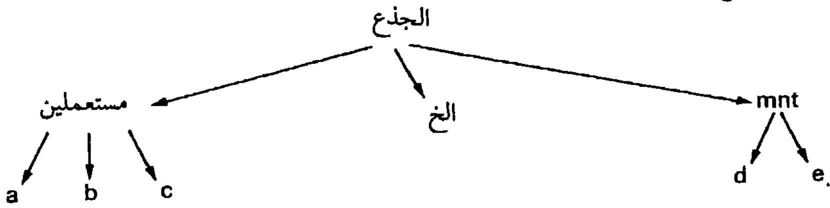
الشكل 2.9 يُوضّح هذه الأوالية . في (a) نجد الشجرية الأولية ، في (b) تنظيم السجلات على الحجم d 333 وفي (c) نتيجة تركيب d 333 على القائمة mnt

في أبجدية UNIX تُدعى واصفات السجلات «inode» ، مجموعة واصفات السجلات هذه هي موجودة في رأس الحجم على شكل جدول من « العقد » (inodes) . هكذا ، عند تشكيل الوحدة الفيزيائية يمكن إنشاء عدة أحجام عليها (الفصل الخامس) . تحتوي واصفات السجلات هذه على لائحة بالفدرات الفيزيائية (تسجيلات فيزيائية)التي تؤلف السجلّ (للسجلات ذات الحجم الكبير ، تحتوي البلوكات الثلاثة الأخيرة بدورها على لائحة بالبلوكات) . السجلات ذات الحجم الصغير يجري تقطيعها بينما السجلات ذات الحجم الكبير تمتاز بتنظيم معقد .

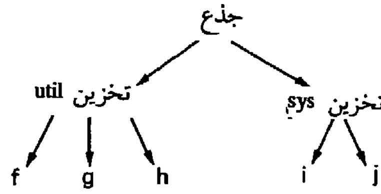
5 . ملقى المستعمل

1.5 . مدخل

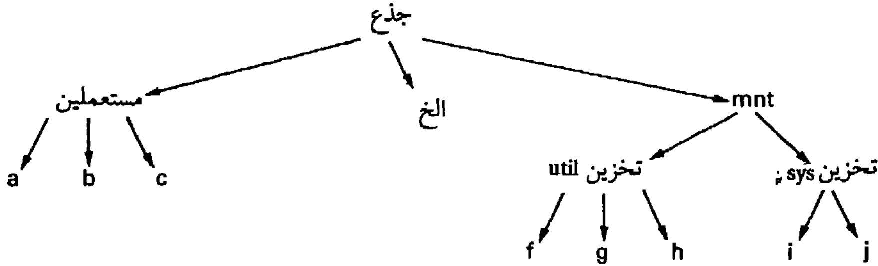
لجهة العدد الكبير من الأوامر الموجودة في النظام UNIX ، فمن غير الممكن عرضها . ولقد رأينا في الفصل الخامس إن الميزة الاساسية للنظام UNIX هي في وجود لغة تحكّم (أوامر) بقوة تعادل لغة للبرمجة مزودة بمفاهيم المتحولة ووجود إنشاءات تحكّم متطورة من نوع «if then else» ، «do while» ، الخ . مفسّر الأوامر هو المهمّة «shell» التي يتمّ إطلاقها عند إعداد المُستعمل بواسطة المهمة «init» من خلال الأمر «login» .



(a) شجرية أولية



(b) سجلات الحجم d333



(c) نتيجة التركيب

الشكل 2.9 : مثل على تركيب الحجم

الشكل (1.9) يمثّل الحالة التي تحتوي على أربع قناصل . القنصلة رقم 4 ليست مستعملة ، المهمة «2–» تدل فقط على إن الأمر «login» يمكن أن يُنفَّذ في كل لحظة . على القنصلة الثالثة نجد أحد المستعملين ، المهمة «init» أنشأت مهمة «sh» مناسبة لمفسِّر الأوامر («Shell») ، والنظام الآن هو جاهز لاستقبال أحد الأوامر من جهة المستعمل . المهمة «sh» هي في « انتظار » رسالة آتية من القنصلة . على القنصلة الثانية ، أطلق المُستعمل المهمة «prog1»، وهذه المهمة جرى إنشاؤها بواسطة مفسِّر الأوامر («sh») الذي يُوضع لاحقاً في « انتظار » إشارة نهاية التنفيذ من جهة المهمة «prog1» . في النهاية وفي حالة القنصلة 1 ، أطلق المستعمل الأمر «ps» (لائحة بالمهام الموجودة ومميّزاتها الأساسية) . ستتمّ تلبية الخدمة المطلوبة بواسطة مهمة «ps» تنشئها المهمة «sh» المُخصَّصة للقنصلة 1 والموضوعة في انتظار إشارة نهاية التنفيذ من «ps» .

2.5 . الخطوط العريضة للغة التحكُّم

بإشراف UNIX، لغة التحكُّم هي عبارة عن لغة للبرمجة حقيقية مزودة بمفاهيم المتغيرات ، المتحوّلات وإنشاءات التحكُّم .

تعرَّف المتحولة بواسطة إسم يتألف من سلسلة من السنمات تبدأ بحرف : مثلاً toto هي متحولة . قيمة المتحولات هي عبارة عن سلاسل من السمات، وتبلغ هذه القيم بجعل إسم المتحولة مسبوقاً بالسنمة « $» . مثلاً ، toto $ يُمثِّل قيمة المتحولة toto . لنفترض البرنامج التالي :

```
toto = nom-utilisateur
ECHO $toto
ECHO toto
```

ستأخذ المتحولة toto القيمة «nom-utilisateur» ، السطر التالي سيعرض إذاً على القتصلة «nom-utilisateur» بينما السطر الأخير سيعرض toto .

من الممكن إستعمال متغيرات حيث يتم تخصيص قيمتها في لحظة تفسير الإجراء المؤلف من عدد من الأوامر .

هذه المتغيرات هي عبارة عن متحولات بالإسم 0, 1, 2 . . . المتحولة 0 تناسب إسم السجل الذي يحتوي على إجراء الأوامر ، والمتحولات 1, 2, 3 . . . تتمتع بقيمة تعادل المتغيرات المضروبة في لحظة الأمر مع المحافظة على ترتيب الأوامر . هذه الأوالية هي شبيهة بالأوالية المستعملة في النظام MS-DOS .

فيما يتعلَّق بإنشاءات التحكم ، فإن shell يحتوي على إنشاء «if then else» ، (IF لائحة من الأوامر THEN لائحة من الأوامر ELSE لائحة من الأوامر FI)، الحلقة «DO FOR» (FOR متحوّلة IN لائحة DO لائحة من الأوامر DONE) ، الحلقة «DO WHILE» (WHILE لائحة الأوامر DO لائحة من الأوامر DONE) ، وكذلك التفريع المتعدّد (CASE كلمة IN متتالية من السمات ( ) لائحة أوامر ;; ESAC) .

3.5 . مفهوم إعادة التوجيه والأنبوب

من السهل كثيراً بإشراف UNIX تعليق سلسلة من الأوامر بسيطة ، هذه الأوامر تتصل فيما بينها إمّا بواسطة أنابيب (pipes) إمّا بواسطة سجلات . من جهة أخرى ، من السهل تحويل هذه السلسلة إلى سلسلة جديدة من الأوامر لا تختلف عن أوامر UNIX بالنسبة للمُستعمل . إضافة إلى الأوامر الأساسية ، فإن النظام UNIX يحتوي أيضاً على مجموعة من الأوامر المُتخصِّصة والمصنوعة من قِبل المسؤولين عن التجهيزات والتركيب .

إحدى الوسائل للاتصال بين الأوامر البسيطة فيما بينها هو توجيه المداخل ـ المخارج . هكذا فكل أمر يحتوي بشكل عام على سجل نموذجي عند الإدخال

وسجل نموذجي عند الإخراج . بالفعل، نبلغ هذه السجلات بواسطة قنوات يمكن أن يُعاد توجيهها نحو سجلات أخرى ، مهما تكن هذه من سجلات نموذجية أو سجلات خاصة مناسبة لوحدات الإدخال ـ الإخراج.

أوامر التوجيه هي «<» لتوجيه الإخراج و«>» لتوجيه الإدخال . مثلاً ، الأمر «II» يسمح بعرض السجلات من القائمة الجارية على القنصلة مع صيغ مميزاتها ، يكفي فقط أن نضرب الأمر التالي :

II > bidon

لكي تظهر على الشاشة جميع المعلومات المخزّنة على السجل bidon . ولو ضربنا الأمر :

more bidon.

فإن المعلومات ستظهر على الشاشة صفحة بعد صفحة لأن الأمر «more» يسمح بعرض السجلات بهذه الطريقة . لقد رأينا في مقدمة هذا الفصل أن المفهوم أنبوب «pipe» يسمح بجعل إنشاء السجل bidon يشكل إنتقالياً وبتصرف المُستعمل . الأمر :

II I more

يسمح بعمل نفس الشيء كالسلسلة المعروضة أعلاه ، إضافة إلى أن المهام المُنشأة لتنفيذ «II » و«more» يمكن أن تدور بشكلٍ متوازٍ مع تشغيل من نوع « مُنتج ـ مستهلك » .

4.5 . التكرار

النقطة الأخرى الواجب إيضاحها من «Shell» هي في وجوب الإشارة إلى أنها تكرارية ، أي باستطاعتنا دعوتها بشكل جلي من داخل السجل الذي يحتوي على سلسلة من الأوامر كي يتم تنفيذ متتالية أخرى . عندما تنتهي المتتالية الأخرى ، تعاود المتتالية الأولى التنفيذ . لدينا إذاً مفهوم شبيه بمفهوم الإجراءات (البرامج الثانوية) في لغات البرمجة ، ويمكن إستعمال متغيرات أو متحولات عامّة (المتحولات غير المصرَّح عنها على أنّها عامة هي متغيرات مركزية في متتالية أوامر معينة) . لجعل السجل الذي يحتوي على لائحة من الأوامر قابلاً للتنفيذ يكفي إستعمال «chmod» التي تسمح بتعديل خاصيات السجل . بالنسبة للمستعمل فإن إسم السجل سيصبح إذاً عبارة عن أمرٍ جديد يؤدي

دوره كأحد أوامر نظام التشغيل .

فلنعاود مثلاً حالة البرنامج الذي نرغب بتنفيذه دورياً . ولقد أعطينا حلاً يستعمل إنشاء تحكُّـم «WHILE DO DONE» ويقوم باختبار متحولة الجواب . الحلُّ الثاني الذي يستعمل التكرارية يمكن أن يكون عبارة عن لائحة الأوامر التـالية المخـزَّنة في السجل «monfich» .

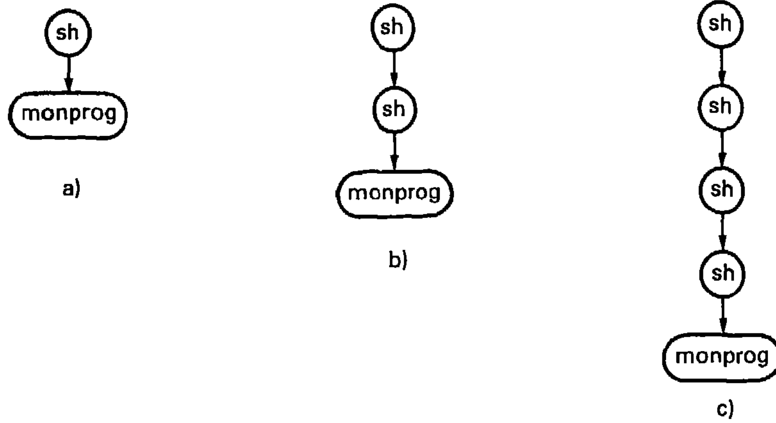
```
ECHO 'execution de : '          ' تنفيذ : '
ECHO $1
$1
ECHO 'voulez vous continuer o/n?'   'هل تريدالاستمرار ؟ '
READ reponse
IF TEST $reponse = o
THEN monfich $1
ELSE ECHO 'arret'               ' توقف '
FI
```

هذه المرَّة فإن إنشاء التحكُّـم هو (IF THEN ELSE FI) حيث FI هي الكلمة المحجوزة في نهاية IF . يُستعمل المتغير 1$ لتمثيل إسم البرنامج الذي نرغب بتنفيذه إذا كان تنفيذ monprog سيتم بضرب

monfich monprog

سلسلة السمات «monprog» سيتم تخصيصها إلـى المتحـولـة «1» . والبرنامج «monprog» سيتم تنفيذه بعد طباعة الرسالة «exécution de monprog» ، لأنه ولتنفيذ أحد البرامج يكفي إعطاء إسمه إلى مُفسِّـر الأوامر (هذا ما يحصل على السطر 3) . بعد ذلك فإن المتحولة «reponse» هي مقروءة ، وإذا كانت قيمتها تعادل السمة «0» فسيتم معاودة تنفيذ لائحة الأوامر «monfich» مع مدخل نفس المتغيرة .

الشكل 3.9 يدل على تنظيم المهام في هكـذا مفهوم . قبـل أي شيء ، يجب ملاحظة أن المهمة «sh» التي هي عبارة عن المفسِّـر تواصل وجودها عندما يطلب أحد المستعملين تنفيـذ أحد البـرامج . مثـلاً ، في الشكـل a.3.9 ، المهمـة «sh» تنشىء المهمة «monprog» المناسبة لتنفيذ البرنامج من نفس الاسم وتوضع في انتظار إشارة نهاية تنفيذ «monprog» قبل السماح للمستعمل بإدخال أمر جديد وتفسيره .



شكل 3.9 : مثل على التكرار

عندما نقوم بتفسير سجل يحتوي على لائحة من الأوامر ، فإن مفسِّر الأوامر يقوم بإنشاء مهمة «sh» تكمن في تفسير السجل ، إضافة لذلك فإذا كان طلب تنفيذ monprog موجوداً في لائحة الأوامر نحصل على الشكل b.3.9 ، المهمة الأولى «sh» هي في انتظار نهاية تفسير السجل ، والثانية هي في نهاية تنفيذ monprog كي تتم متابعة تفسير لائحة الأوامر .

بشكل عام ، وفي كل مرة نُفسِّر فيها لائحة الأوامر ، بشكل متكرِّر، يجري إنشاء مهمة تفسير جديدة للأوامر «sh» ، وفي التنفيذ الثالث للبرنامج monprog ، سنحصل على المخطط المُمثَّل على الشكل c.3.9 . كل مهمة «sh» هي في انتظار اشارة نهاية تنفيذ المهمة الابنة ، إشارة يتم إرسالها بواسطة مفسِّر الأوامر عندما ينتهي تفسير السجل monfich . المهمة الأولى «sh» المُناسبة لمفسِّر الأوامر هي في انتظار أمر من القنصلة .

5.5 . البرامج المُساعدة الأخرى

ضمن الأوامر الأساسية نجد أوامر متطورة لمعالجة السجلات تسمح باستخراج الأسطر ، ومقارنة مضمون سجلّين (البحث عن الفروقات بين صيغتين) ، والبحث عن سلاسل السمات في السجل ، الخ . هذه الأوامر تُسهِّل بشكل كبير عمل المهندس المسؤول عن التنظيم .

من جهة أخرى ، فإن المهمة «mail» تقدم نظام نقل إلكتروني يسمح للمستعملين بتبادل الرسائل فيما بينهم . توضع الرسائل في علب للرسائل مما يسمح

للموجَّه إليه ببلوغها عندما يرغب في ذلك . عندما يقوم أحد المستعملين بتنفيذ الأمر «login» لاعداد دوام عمل معيّن (session job) ، فإنّ النظام يُنبِّهه فيما إذا كانت الرسالة قد أُرسلت إليه في حال غيابه . يمكن للرسائل ليس فقط أن تُقرأ ولكن يمكن أيضاً أن يتم تحويلها إلى سجلات للأرشيف والعرض ,والتعديل, الخ .بواسطة المُستعملين .

فهرست

1990 / 3 / 33

هذا الكتاب

يعرض هذا الكتاب المبادىء العامة لأنظمة تشغيل الحاسبات الإلكترونية ، ويتناول أهم النظم المستعملة(« متعدِّدة ـ المستعملين » ، « مُتعدِّدة المهام » ، « متعددة ـ البرامج » ، « تجزئة الوقت ») ، ويشرح دور المرقاب في جميع النظم المذكورة ، والطرق الأساسية المستعملة في تنظيم الذاكرة ووحدات الإدخال ـ الإخراج ، إضافة إلى تنظيم عمل المُعالج المركزي في خدمة جميع المهام .

وفي النهاية يستعرض هذا الكتاب أهم النظم المُستعملة في الميكروحاسبات : UNIX , IRM 86, MS-DOS, CP /M .

المؤسسة الجامعية للدراسات والنشر والتوزيع